APLIKASI KRIPTOGRAFI SEBAGAI PENGAMANAN DATA PADA FILE DENGAN ALGORITMA BLOWFISH

TUGAS AKHIR
Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

Oleh :
RUDI YULISMAN
10351022937

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2009

APLIKASI KRIPTOGRAFI SEBAGAI PENGAMANAN DATA PADA FILE DENGAN ALGORITMA BLOWFISH

RUDI YULISMAN

10351022937

Tanggal Sidang : 8 Desember 2009

Periode Wisuda : Februari 2010

Jurusan Teknik Informatika

Fakultas Sains dan Teknologi

Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Kerahasiaan data sangat diperlukan baik dalam suatu organisasi maupun pribadi, bahkan dalam jaringan internasional yang sering disebut internet. Untuk menjaga kerahasiaan data, maka diperlukan metode enkripsi agar data tidak dapat dibaca dan dimengerti.

Algoritma blowfish sebagai penyandi data yang cepat dan menggunakan 64 *bit* untuk setiap 16 kali putaran secara berulang-ulang, desainnya mudah untuk dianalisa yang membuatnya tahan terhadap kesalahan dan akan berada pada *domain public* (bebas paten).

File hasil enkripsi pada aplikasi menggunakan format *.blw. Mengalami penambahan waktu pada ukuran file yang berbeda. Hasil pengujian keamanan pada file enkripsi menggunakan metode *brute force attack* menunjukan hasil bahwa dari 50 kali usaha pembobolan *password*, persentase kegagalan 100 %. Hasil dari ujicoba terhadap berbagai jenis file menunjukkan bahwa aplikasi dapat digunakan untuk semua tipe file tanpa ada kesalahan.

Kata kunci : Algoritma Blowfish, Dekripsi, Enkripsi, File, Key, Kriptografi.

CRYPTOGRAPHY APPLICATIONS TO DATA SECURITY AS FILE BLOWFISH ALGORITHM

RUDI YULISMAN

10351022937

Date of Final Exam : *December 08 th, 2009*

Graduation Cremony Priod : February 2010

Informatics Engineering Departement

Faculty of Sciences and Technology

State Islamic University of Sultan Syarif Kasim Riau

ABSTRACT

Confidentiality of data is needed both within an organization or individual, even in an international network that is often called the Internet. To maintain the confidentiality of data, the encryption method is required in order not to be read and understood.

Blowfish algorithm for encoding data quickly and use 64 bits for each round 16 times repeatedly, the design is easy to be analyzed is made resistant to errors and will be in the public domain (free patent).

*Encryption of the file format *. blw on the application and the time required for encryption and decryption of large files. Show differences in safety test results on file encryption using brute-force attack method results showed that of 50 break-password attempts, the percentage of failures 100% . Results of the study showed that the program can be used for all types of files without any errors.*

Keywords : Algorithm Blowfish, Cryptography, Decryption , Encryption, File, Key.

DAFTAR ISI

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemakaian teknologi komputer sebagai salah satu aplikasi dari teknologi informasi sudah menjadi suatu kebutuhan, karena banyak pekerjaan yang dapat diselesaikan dengan cepat, akurat, dan efisien. Dengan berkembangnya teknik telekomunikasi dan sistem pengolahan data yang berkaitan erat dengan komunikasi antar pengguna komputer yang satu dengan komputer yang lain sehingga masalah keamanan merupakan salah satu aspek penting dari suatu sistem informasi, khususnya komunikasi yang menggunakan komputer dan terhubung ke jaringan.

Seperti yang kita lihat saat ini, internet telah berkembang semakin banyak dan terkoneksi di berbagai penjuru dunia. Dari hari ke hari informasi yang terkandung di dalam jaringan internet tersebut semakin lengkap, akurat dan penting. Manajemen dan perlindungan keamanan masing-masing jaringan diserahkan sepenuhnya kepada penanggungjawab jaringan (administrator jaringan internet). Salah satu cara untuk mendapatkan ID milik orang lain, seorang cracker berusaha mencuri *file* password dari suatu sistem, kemudian menganalisanya dan cracker secara pribadi berusaha mencuri *file* rahasia suatu perusahaan untuk dijual ke perusahaan lawan.

Sejalan dengan perkembangan teknologi informasi tersebut, untuk pengamanan data yang melewati jaringan terbuka seperti internet, tidak ada jalan

lain selain penggunaan enkripsi sehingga data yang lewat tidak bisa dimanfaatkan orang yang tidak berhak ataupun oleh cracker. Demi menjamin keamanan ini, telah diciptakan suatu metode penyandian (*encryption* dan *decryption*) terhadap data yang ingin dilindungi. Data yang dienkripsi tidak akan dapat dibaca oleh orang yang tidak berhak. Oleh karena itu, telah banyak pengembangan-pengembangan yang dilakukan untuk membentuk algoritma enkripsi dan dekripsi (algoritma kriptografi) yang cepat dan aman.

Kriptografi merupakan ilmu dan seni untuk menjaga kerahasiaan data dan keaslian data tersebut. Dalam pengamanan data tidak hanya sebatas data tersebut tidak dapat dibaca orang lain, tetapi juga bagaimana agar data tersebut tidak dapat diubah atau dimodifikasi, sehingga dibutuhkan suatu cara untuk memastikan keaslian dari data yang dikirim tersebut. Jenis kriptografi terdiri dari *asymmetric algorithms* yaitu algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya dan *symmetric algorithms* yaitu algoritma yang menggunakan kunci yang sama untuk proses enkripsi dan dekripsinya. Sebagian algoritma yang menggunakan *asymmetric algorithms* adalah DSA, RSA, LUC sedangkan algoritma yang menggunakan *symmetric algorithms* adalah DES, RC6, IDEA, Blowfish.

Symmetric algorithms dibagi menjadi *stream chipper* yaitu algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk *bit* tunggal, pada rangkaian *bit* ini dienkripsikan dan didekripsikan *bit* per *bit* dan *block chipper* yaitu algoritma kriptografi beroperasi pada plainteks/cipherteks dalam bentuk

blok *bit*, yang dalam hal ini rangkaian *bit* dibagi menjadi blok-blok *bit* yang panjangnya sudah ditentukan sebelumnya.

Algoritma blowfish menggunakan kunci yang sama dan *variable* saat proses enkripsi dan dekripsi, oleh karena itu pengirim data harus bisa mengirimkan sebuah kunci kepada sipenerima tanpa diketahui oleh pihak yang lain dengan kesepakatan mereka berdua. Algoritma blowfish termasuk kedalam *block chipper* karena menggunakan 64 *bit* untuk setiap 16 kali putaran secara berulang-ulang. Kecepatan blowfish menyandi data adalah 32 *bits microprocessors* pada rentang dari 18 *clock cycles* per *byte*, bisa berjalan kurang dari 5k dari *memory* dan merupakan jenis penyandian yang mudah untuk diimplementasikan selain itu algoritma blowfish bebas paten dan akan berada pada *domain public* dan sampai kini belum ada *cryptanalysis* yang dapat membongkar pesan tanpa kunci dari enkripsi oleh blowfish.

Dalam tugas akhir ini penulis akan membahas lebih lanjut tentang aplikasi kriptografi dengan algoritma blowfish yang memberikan keamanan pada *file*, sehingga *file* tersebut dapat di enkripsi dan dekripsi menjadi data yang bersifat rahasia.

1.2 Rumusan Masalah

Sebagaimana telah dipaparkan sebelumnya pada latar belakang, maka didapatkan rumusan masalah dari tugas akhir ini yaitu bagaimana membangun suatu aplikasi kriptografi yang dapat mengenkripsi dan dekripsi file dengan menggunakan algoritma blowfish.

1.3 Batasan Masalah

Adapun batasan masalah dalam tugas akhir ini adalah sebagai berikut :

1. Ukuran kunci yang dipakai 32 *bit* untuk beberapa *array* subkunci (*subkey*).
2. Pengujian keamanan pada file enkripsi menggunakan metode *brute force attack.*

1.4 Tujuan

Tujuan yang ingin dicapai dalam penyusunan tugas akhir ini adalah :

1. Memahami kriptografi dengan menggunakan algoritma blowfish.
2. Membangun sebuah aplikasi kriptografi untuk enkripsi dan dekripsi *file*.

1.5 Sistematika Penulisan

Sistematika penulisan dalam penyusunan laporan tugas akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan dasar-dasar dari penulisan laporan tugas akhir, yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan, serta sistematika penulisan laporan tugas akhir.

BAB II LANDASAN TEORI

Bab ini membahas teori-teori yang berhubungan dengan topik penelitian, meliputi keamanan data, kriptografi dan algoritma blowfish.

BAB III METODOLOGI PENELITIAN

Bab ini membahas tentang metodologi yang digunakan dalam penelitian dan pengembangan perangkat lunak.

BAB IV ANALISIS DAN PERANCANGAN

Pada bab ini merupakan pembahasan tentang analisis perangkat lunak, meliputi analisis, analisis masalah, analisis metode, analisis kebutuhan sistem, serta perancangan. Perancangan sistem yang terdiri dari perancangan diagram alir (*flowchart*).

BAB V IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas implementasi dan pengujian yang dilakukan terhadap Aplikasi kriptografi sebagai pengamanan data pada *file* dengan algoritma blowfish.

BAB VI PENUTUP

Bab ini berisi kesimpulan yang dihasikan dari pembahasan tentang Aplikasi pengamanan data pada *file* dengan algoritma blowfish dan saran sebagai hasil akhir dari penelitian yang telah dilakukan.

BAB II

LANDASAN TEORI

Landasan teori disusun berdasarkan teori-teori yang berhubungan dengan keamanan data, kriptografi dan algoritma blowfish.

2.1 Keamanan Data

Keamanan data biasanya terkait hal-hal berikut:(Setiadi, 2004)

a. **Fisik,** dalam hal ini pihak yang tidak berwenang terhadap data berusaha mendapatkan data dengan melakukan kegiatan sabotase atau penghancuran tempat penyimpanan data.
b. **Organisasi,** dalam hal ini pihak yang tidak berwenang untuk mendapatkan data melalui kelalaian atau kebocoran anggota yang menangani data tersebut.
c. **Ancaman dari luar,** dalam hal ini pihak yang tidak berwenang berusaha untuk mendapatkan data melalui media komunikasi dan juga melakukan pencurian data yang tersimpan di dalam komputer.

Fungsi keamanan komputer adalah menjaga tiga karakteristik, yaitu:

a. ***Secrecy***, adalah isi dari program komputer hanya dapat diakses oleh orang yang berhak. Tipe yang termasuk di sini adalah *reading, viewing, printing,* atau hanya yang mengetahui keberadaan sebuah objek.
b. ***Integrity***, adalah isi dari komputer yang dapat dimodifikasi oleh orang yang berhak, yang termasuk disini adalah *writing, changing status, deleting,* dan *creating*.

c. ***Availability***, adalah isi dari komputer yang tersedia untuk beberapa kelompok yang diberi hak.

Keamanan data merupakan hal penting, karena memiliki peranan yang sssangat besar dalam sistem komputer, jaringan komputer, dan penggunaan teknologi komputer.

Dalam mengatasi masalah keamanan data tersebut, menurut *International Telecomunication Union-Telecomunication Standarization Sector (ITU-T)* yang merekomendasikan X.8000 ada beberapa teknik dalam menjaga keamanan dan kerahasiaan dalam sistem komputer, jaringan komputer, dan *internet*, yaitu: (Raharjo,1998).

2.1.1 *Privacy*

Ketika sebuah pesan atau informasi dinilai sensitif perlu mendapat perlindungan. Data-data yang dikirim seperti data pribadi atau alamat *website* tertentu dapat disalahgunakan oleh *user* yang tidak bertanggung jawab, misalnya informasi yang diberikan saat mendaftar *e-mail* gratis atau *mailing list* maupun *newsgroup* dapat terekam di *database* situs tersebut. Informasi ini bisa saja digunakan oleh orang lain untuk penawaran suatu produk tertentu, belum lagi serbuan dari *spyware* yang biasanya terdapat pada perangkat lunak *freeware* atau gratis. Dengan kriptografi data rahasia akan terkunci dan tidak dapat dibuka oleh pihak lain.

2.1.2 *Integrity*

Integritas data diperlukan untuk menjamin bahwa data yang dikirim harus benar-benar data asli yang dikirim oleh pengirim atau *user* yang benar-benar

mengirimkannya. Selain itu integritas harus dapat memberikan jaminan untuk tiap bagian bahwa pesan tidak mengalami perubahan dari saat dibuat hingga dibuka. Untuk menjaga agar data yang dikirim utuh dan asli perlu dilakukan teknik kriptografi.

2.1.3 *Authencity*

Hal penting dalam keamanan adalah keaslian, pembuktian pesan, *file*, dokumen ataupun data lainnya yang dikatakan autentik dan berasal dari sumber resmi. Umumnya ada 3 pendekatan dalam pembuktian suatu pesan. Pertama, *user* memberikan suatu kata kunci yang hanya diketahui *user* tersebut sebagai *password*, *PIN* atau identitas lainnya. Kedua, penggunaan *peripheral* untuk pembuktian seperti tanda tangan *digital*. Ketiga, dengan cara menguji semua identitas yang mewakili *user* tersebut seperti suara, sidik jari, retina mata, dan lainnya.

2.1.4 *Non-Repudiation*

Hal ini berkaitan dengan penyangkalan apa yang telah dilakukan oleh seseorang yang melakukan sesuatu, namun tidak dapat dibuktikan oleh penyangkal. Misalkan seorang *user* yang menyangkal telah memesan suatu barang atau telah menerima paket barang.

2.1.5 *Encryp*

Enkripsi adalah sebuah proses dimana sebuah pesan ditransformasikan ke bentuk pesan lain (*chipertext*) menggunakan fungsi matematis dan sebuah enkripsi *password* spesial yang dikenal dengan istilah *key*. Keuntungan penggunaan enkripsi adalah bila metode lain untuk melindungi *data* (daftar kontrol akses, *file*,

password) berhasil dicuri atau dibongkar penyusup, ia harus mengeluarkan tenaga ekstra dan sumber daya lain untuk dapat membaca atau mendeskripsikan data tersebut.

2.2 Kriptografi

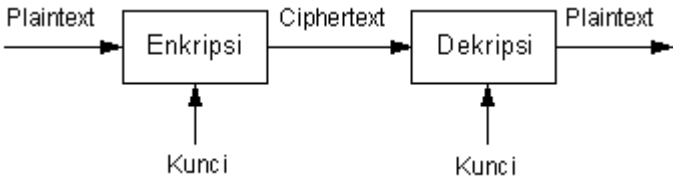
Kriptografi dipergunakan secara terbatas oleh bangsa Mesir 4000 tahun lalu. Kriptografi berasal dari dua kata yaitu *Crypto* dapat diartikan rahasia (*secret*) dan *graphy* dapat diartikan tulisan (*writing*) jadi Kriptografi dapat diartikan sebagai suatu ilmu atau seni untuk mengamankan pesan agar tidak diketahui oleh pihak yang tidak diinginkan atau kriptografi adalah seni dari penulisan rahasia atau membaca sandi atau tulisan–tulisan rahasia.(munir, 2007)

Layanan-layanan yang disediakan oleh kriptografi antara lain:

1. Kerahasiaan (*confidentiality*), menyediakan privasi untuk pesan dan menyimpan data dengan menyembunyikannya.
2. Integritas pesan (*message integrity*), menyediakan jaminan untuk semua pihak bahwa pesan tidak dimodifikasi secara ilegal.
3. Tidak adanya penyangkalan (*non repudiation*), dapat membuktikan bahwa dokumen memang benar datang misalnya X dan X tidak menyangkalnya.
4. Autentikasi (*authentication*), yaitu dengan mengidentifikasi asal dari pesan, dan memeriksa identitas dari pengguna sistem komputer.

Dalam kriptografi terdapat tulisan yang tidak dapat dimengerti apa maksudnya karena terdapat proses yang disebut dengan enkripsi dan dekripsi. Enkripsi dapat didefinisikan sebagai proses konversi suatu informasi dari bentuk yang dapat dibaca ke dalam bentuk yang tidak dapat dimengerti oleh pihak lain.

Kode hasil enkripsi disebut dengan *chiper*, sedangkan proses membalikkan *chiper* menjadi bentuk yang dapat dimengerti disebut dengan dekripsi.



Gambar 2.1 Proses Enkripsi dan Dekripsi

2.2.1 Sejarah Kriptografi

Kriptografi sudah digunakan sekitar 40 abad yang lalu oleh orang-orang Mesir untuk mengirim pesan ke pasukan yang berada di medan perang dan agar pesan tersebut tidak terbaca oleh pihak musuh walaupun pembawa pesan tersebut tertangkap oleh musuh.

Sekitar 400 SM, kriptografi digunakan oleh bangsa Spartan dalam bentuk sepotong papirus atau perkamen yang dibungkus dengan batang kayu. Pada zaman Romawi kuno, ketika Julius Caesar ingin mengirimkan pesan rahasia pada seorang Jendral di medan perang. Pesan tersebut harus dikirimkan melalui seorang prajurit, tetapi karena pesan tersebut mengandung rahasia, Julius Caesar tidak ingin pesan tersebut terbuka di tengah jalan.

Di sini Julius Caesar memikirkan bagaimana mengatasinya yaitu dengan mengacak isi pesan tersebut menjadi suatu pesan yang tidak dapat dipahami oleh siapapun kecuali hanya dapat dipahami oleh Jendralnya saja. Tentu sang Jendral telah diberi tahu sebelumnya bagaimana cara membaca pesan yang teracak tersebut, karena telah mengetahui kuncinya.

Pada perang dunia kedua, Jerman menggunakan mesin *enigma* atau juga disebut dengan mesin rotor yang digunakan Hitler untuk mengirim pesan kepada tentaranya di medan perang. Jerman sangat percaya bahwa pesan yang dienkripsi menggunakan *enigma* tidak dapat dipecahkan. Tapi anggapan itu keliru, setelah bertahun-tahun sekutu mempelajarinya dan berhasil memecahkan kode-kode tersebut.

Setelah Jerman mengetahui bahwa *enigma* dapat dipecahkan, maka enigma mengalami beberapa kali perubahan. *Enigma* yang digunakan Jerman dapat mengenkripsi suatu pesan sehingga mempunyai 15×1018 kemungkinan untuk dapat mendekripsi pesan. Perkembangan komputer dan sistem komunikasi pada tahun 60-an berdampak pada permintaan dari pihak-pihak tertentu sebagai sarana untuk melindungi informasi dalam bentuk digital dan untuk menyediakan layanan keamanan. Dimulai dari usaha Feistel dari IBM di awal tahun 70-an dan mencapai puncaknya pada 1977 dengan pengangkatan *DES* (*Data Encryption Standard*) sebagai standar pemrosesan informasi federal Amerika Serikat untuk mengenkripsi informasi yang tidak belum diklasifikasi. DES merupakan mekanisme kriptografi yang paling dikenal sepanjang sejarah.

Pengembangan paling mengejutkan dalam sejarah kriptografi terjadi pada 1976 saat Diffie dan Hellman mempublikasikan ”*New Directions in Cryptography*”. Tulisan ini memperkenalkan konsep revolusioner kriptografi kunci publik dan juga memberikan metode baru untuk pertukaran kunci, keamanan yang berdasar pada kekuatan masalah algoritma diskrit. Meskipun Diffie dan Hellman tidak memiliki realisasi praktis pada ide enkripsi kunci publik

saat itu, idenya sangat jelas dan menumbuhkan ketertarikan yang luas pada komunitas kriptografi.

Pada 1978 Rivest, Shamir dan Adleman menemukan rancangan enkripsi kunci publik yang sekarang disebut RSA. Rancangan RSA berdasar pada masalah faktorisasi bilangan yang sulit, dan menggiatkan kembali usaha untuk menemukan metode yang lebih *efisien* untuk pemfaktoran. Tahun 80-an terjadi peningkatan luas di area ini, sistem RSA masih aman. Sistem lain yang merupakan rancangan kunci publik ditemukan oleh Taher ElGamal pada tahun 1985.

Rancangan ini berdasar pada masalah algoritma diskrit. Salah satu kontribusi penting dari kriptografi kunci publik adalah tanda tangan digital. Pada 1991 standar internasional pertama untuk tanda tangan digital diadopsi. Standar ini berdasar pada rancangan kunci publik RSA. Pada 1994 pemerintah Amerika Serikat mengadopsi *Digital Signature Standard*, sebuah mekanisme kriptografi yang berdasar pada algoritma ElGamal.

Sistem kriptografi atau *Cryptosystem* adalah sebuah algoritma kriptografi ditambah semua kemungkinan *plaintext, chipertext* dan kunci. Dalam sistem ini, seperangkat parameter yang menentukan transformasi pen-*chiper*-an tertentu disebut suatu set kunci. Proses enkripsi dan dekripsi diatur oleh satu atau beberapa kunci kriptografi.

Dalam menjaga kerahasiaan data, kriptografi mentransformasikan data jelas (*plaintext*) ke dalam bentuk data sandi (*chipertext*) yang tidak dapat dikenali. *Chipertext* inilah yang kemudian dikirimkan oleh pengirim (*sender*) kepada

penerima (*receiver*). Setelah sampai di penerima, *chipertext* tersebut ditranformasikan kembali ke dalam bentuk *plaintext* agar dapat dikenali.

Proses tranformasi dari *plaintext* menjadi *chipertext* disebut proses *Enchiperment* atau enkripsi (*encryption*), sedangkan proses mentransformasikan kembali *chipertext* menjadi *plaintext* disebut proses dekripsi (*decryption*). Untuk mengenkripsi dan mendekripsi data. Kriptografi menggunakan suatu algoritma (*chiper*) dan kunci (*key*). *chiper* adalah fungsi matematika yang digunakan untuk mengenkripsi dan mendekripsi data. Sedangkan kunci merupakan sederetan *bit* yang diperlukan untuk mengenkripsi dan mendekripsi data.

Algoritma kriptografi modern tidak lagi mengandalkan keamanannya pada kerahasiaan algoritma tetapi kerahasiaan kunci. *Plaintext* yang sama bila disandikan dengan kunci yang berbeda akan menghasilkan *chipertext* yang berbeda pula. Dengan demikian algoritma kriptografi dapat bersifat umum dan boleh diketahui oleh siapa saja, akan tetapi tanpa pengetahuan tentang kunci, data tersandi tetap saja tidak dapat terpecahkan.

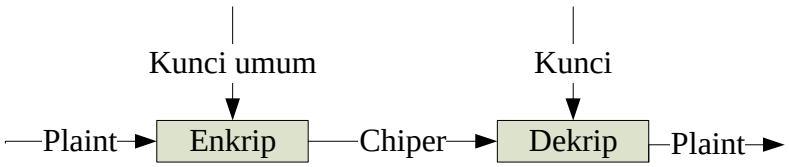
2.2.2 Jenis Algoritma Kriptografi

Berdasarkan kunci yang dipakai, algoritma kriptografi dapat dibedakan atas dua golongan, yaitu :

2.2.2.1 *Asymmetric Algorithms*

Asymmetric Algorithms adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Algoritma ini disebut juga algoritma kunci umum (*public key algorithm*) karena kunci untuk enkripsi dibuat

umum (*public key*) atau dapat diketahui oleh setiap orang, tapi kunci untuk dekripsi hanya diketahui oleh orang yang berwenang mengetahui data yang disandikan atau sering disebut kunci pribadi (*private key*). Proses enkripsi-dekripsi algoritma asimetris dapat dilihat pada gambar dibawah ini :



Gambar 2.2 Proses Enkripsi dan Dekripsi Algoritma Asimetris

Pada algoritma *public key* ini, semua orang dapat mengenkripsi data dengan memakai *public key* penerima yang telah diketahui secara umum. Akan tetapi data yang telah terenkripsi tersebut hanya dapat didekripsi dengan menggunakan *private key* yang hanya diketahui oleh penerima.

2.2.2.1.1 DSA

Pada bulan Agustus 1991, NIST (*The National Institute of Standard and Technology*) mengumumkan algoritma sidik dijital yang disebut *Digital Signature Algorithm* (DSA). DSA dijadikan sebagai standar dari *Digital Signature Standard* (DSS).

DSA tidak dapat digunakan untuk enkripsi. DSA mempunyai dua fungsi utama:

1. Pembentukan sidik digital (*signature generation*)
2. Pemeriksaan keabsahan sidik digital (*signature verification*).

DSA menggunakan dua buah kunci, yaitu kunci publik dan kunci rahasia. Pembentukan sidik digital menggunakan kunci rahasia pengirim, sedangkan verifikasi sidik digital menggunakan kunci publik pengirim. DSA menggunakan fungsi *hash* SHA (*Secure Hash Algorithm*) untuk mengubah pesan menjadi *message digest* yang berukuran 160 *bit*.

2.2.2.1.2 RSA

Algoritma RSA dibuat oleh 3 orang peneliti dari MIT (*Massachussets Institute of Technology*) pada tahun 1976, yaitu: Ron (R)ivest, Adi (S)hamir, dan Leonard (A)dleman.

RSA yang menggunakan algoritma asimetris mempunyai dua kunci yang berbeda, disebut pasangan kunci (*key pair*) untuk proses enkripsi dan dekripsi. Kunci-kunci yang ada pada pasangan kunci mempunyai hubungan secara matematis, tetapi tidak dapat dilihat secara komputasi untuk mendeduksi kunci yang satu ke pasangannya. Algoritma ini disebut kunci publik, karena kunci enkripsi dapat disebarkan. Orang-orang dapat menggunakan kunci publik ini, tapi hanya orang yang mempunyai *private key* sajalah yang bisa mendekripsi data tersebut.

Dalam pemakaian sistem penyandian RSA dalam kehidupan sehari-hari adalah *signature* atau tanda tangan *digital* dalam surat elektronik dan untuk autentikasi sebuah data. Untuk meyakinkan penerima surat elektronik yang ditandatangani, diperlukan pembuktian bahwa surat elektronik tersebut memang berasal dari sipengirim.

RSA dalam prosesnya tergolong lambat dibandingkan dengan metode enkripsi yang popular baik metode yang menggunakan kunci simetris, dimana metode lain lebih cepat.(munir, 2007)

2.2.2.1.3 LUC

Algoritma LUC merupakan algoritma kriptografi kunci publik yang dikembangkan oleh Peter J. Smith dari New Zealand pada tahun 1993. Metode LUC ini dirancang oleh Peter J. Smith setelah ia berhasil meneliti dan melihat kelemahan dari metode RSA. Metode LUC ini menggunakan fungsi Lucas yang dapat menutupi kelemahan metode RSA yang menggunakan fungsi pangkat. Kemungkinan untuk menjebol RSA menjadi ada karena masalah pangkat tersebut. Fungsi Lucas ini dapat mencegah kemungkinan tersebut.

Algoritma LUC sendiri hampir sama dengan metode RSA hanya saja fungsi pangkat pada metode RSA diganti dengan fungsi Lucas pada metode LUC. Metode LUC ini menggunakan dua buah bilangan prima besar, p dan q untuk menghasilkan pasangan kunci publik, e dan n, dimana n = p * q, serta *private key*, d. Sedangkan nilai p dan q sendiri tidak digunakan dalam algoritma ini.

Algoritma ini juga menggunakan Extended Euclidean algorithm untuk menghasilkan nilai kunci dekripsi d dari kunci enkripsi e. Nilai kunci enkripsi e merupakan sebuah bilangan acak yang dibangkitkan sedemikian rupa sehingga harus relatif prima terhadap p – 1, q – 1, p + 1 dan q + 1. Sedangkan panjang plaintext harus lebih kecil daripada n.

2.2.2.2 *Symmetric Algorithms*

Algoritma kriptografi simetris atau disebut juga algoritma kriptografi konvensioanl adalah algoritma yang menggunakan kunci untuk proses enkripsi sama dengan kunci untuk proses dekripsi.

Algoritma kriptografi simetri dapat dikelompokkan menjadi dua kategori, yaitu:

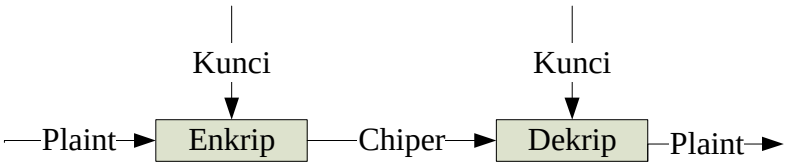
1. ***Chiper* aliran (s*tream chiper*)**

Algoritma kriptografi beroperasi pada plainteks/chiperteks dalam bentuk *bit* tunggal, pada rangkaian *bit* ini dienkripsikan dan didekripsikan *bit* per *bit*.

2. ***Chiper* blok (*block chiper*)**

Algoritma kriptografi beroperasi pada plainteks/chiperteks dalam bentuk blok *bit*, yang dalam hal ini rangkaian *bit* dibagi menjadi blok-blok *bit* yang panjangnya sudah ditentukan sebelumnya.

Proses enkripsi dan dekripsi algoritma kriptografi simetris dapat dilihat pada gambar dibawah ini :



Gambar 2.3 Proses Enkripsi dan Dekripsi Algoritma Simetris

2.2.2.2.1 Algoritma DES

Algoritma DES merupakan kunci simetris, dikembangkan di IBM dibawah kepemimpinan W.L. Tuchman pada tahun 1972. Algoritma ini didasarkan pada

algoritma LUCIFER yang dibuat oleh Horst Feistel. Algoritma DES telah mendapat persetujuan dari *National Bureau of Standard* (NBS) setelah penilaian kekuatannya oleh *National Security Agency* (NSA) Amerika Serikat.

DES beroperasi pada ukuran blok 64 *bit*. DES mengenkripsikan 64 *bit* plainteks menjadi 64 *bit* chiperteks dengan menggunakan 56 *bit* kunci internal (*internal key*) atau upa kunci (*subkey*). Kunci internal dibangkitkan dari kunci eksternal (*external key*) dengan panjang kunci eksternal DES hanya 64 bit atau 8 karakter, itupun yang dipakai hanya 56 bit. Pada rancangan awal, panjang kunci yang diusulkan IBM adalah 128 bit, tetapi atas permintaan NSA, panjang kunci diperkecil menjadi 56 bit, maka DES memiliki kunci yang lemah dan desain struktur internal DES dimana bagian subsitusinya (S-box) masih dirahasiakan.(safitri, 2007)

2.2.2.2.2 Algoritma RC 6

Algoritma RC6 merupakan salah satu kandidat *Advanced Encryption Standard* (AES) yang diajukan oleh RSA *Security Laboratories* kepada NIST. Dirancang oleh Ronald L Rivest, M.J.B. Robshaw, R. Sidney dan Y.L. Yin, algoritma ini merupakan pengembangan dari algoritma sebelumnya yaitu RC5 dan telah memenuhi semua kriteria yang diajukan oleh NIST.

RC6 adalah algoritma kriptografi *symmetric key* yang merupakan perkembangan dari algoritma RC5 yang disertakan dalam *Advances Encrytion Standard* (AES). RC6 menggunakan esensi dari *data dependent rotations*. RC6 adalah algoritma yang menggunakan ukuran blok hingga 128 *bit*, dengan ukuran kunci yang digunakan bervariasi antara 128, 192 dan 256 *bit*. RC6 memecah blok

128 *bit* menjadi 4 buah blok 32-bit RC6 mengikuti aturan enam operasi dasar yaitu : penjumlahan, pengurangan, *exclusive* OR, perkalian bilangan *integer*, geser *bit* kekanan, dan geser *bit* ke kiri.

RC6 terdiri dari dua jaringan Feistel dimana datanya dicampur dengan rotasi yang bergantung pada isi data tersebut, sehingga membutuhkan waktu untuk proses pengenkripsian data apalagi data tersebut berukuran besar.(rudianto, 2008)

2.2.2.2.3 Algoritma IDEA

Algoritma IDEA (*International Data Encryption Algorithm*) merupakan sebuah algoritma kunci simetrik (*Symmetric Algorithms*). IDEA muncul pertama kali pada tahun 1990 yang dikembangkan oleh ilmuwan Xueijia Lai dan James L Massey.

IDEA mengenkripsi plaintext menjadi chiperteks yang lemah hanya mengalami 8 putaran. Algoritma ini membagi plaintext yang akan dienkripsi menjadi 4 blok, masing-masing terdiri dari 16 *bit*, tetapi IDEA lebih meningkatkan proses keamanan kunci, dimana 52 upa kunci (sub-keys) yang terdiri dari 16 *bit* dibangkitkan dari kunci utama (master key) yang terdiri 128 *bit*. Lalu pada setiap putarannya digunakan 6 kunci. Setelah itu dilakukan transformasi final dengan 4 kunci untuk membalikkan posisi ke operasi dasar. Algoritma IDEA mempunyai paten untuk mencegah pembajakan dan kejahatan.(Jethefer, 2006).

2.3 Algoritma blowfish

Blowfish merupakan sebuah algoritma kunci simetrik (*symmetric Algorithms)* chiper blok yang dirancang pada tahun 1993 oleh Bruce Schneier.

Pada saat itu banyak sekali rancangan algoritma yang ditawarkan, namun hampir semua terhalang oleh paten atau kerahasiaan pemerintah Amerika. Schneier menyatakan bahwa blowfish bebas paten dan akan berada pada domain publik. Dengan pernyataan Schneier tersebut blowfish telah mendapatkan tempat di dunia kriptografi, khususnya bagi masyarakat yang membutuhkan algoritma kriptografi yang cepat, kuat, dan tidak terhalang oleh lisensi.

Keberhasilan blowfish dalam menembus pasar telah terbukti dengan diadopsinya blowfish sebagai *Open Cryptography Interface* (OCI) pada *kernel Linux versi* 2.5 keatas. Dengan diadopsinya blowfish, maka telah menyatakan bahwa dunia *open source* menganggap blowfish adalah salah satu algoritma yang terbaik.

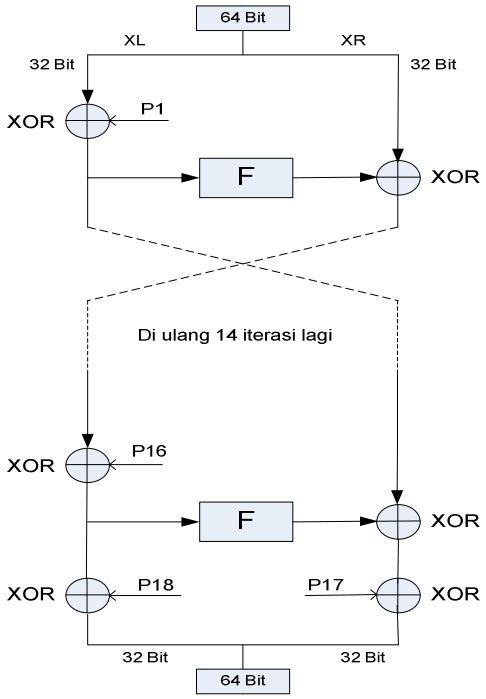
Blowfish adalah algoritma yang menerapkan jaringan Feistel (*Feistel Network*) yang terdiri dari 16 putaran. Blowfish merupakan chiper blok 64 *bit* dengan panjang kunci variabel. Algoritma ini terdiri dari dua bagian: *key expansion* dan enkripsi data.

Enkripsi data terdiri dari iterasi fungsi sederhana sebanyak 16 kali putaran. Setiap putaran terdiri dari permutasi kunci *dependent* dan substitusi kunci dan data *dependent*. Operasi berbentuk penambahan dan XOR pada variable 32 *bit*. Tambahan operasi lainnya hanya empat penelusuran tabel (*table lookup*) *array* berindeks untuk setiap putaran.

Blowfish menerapkan teknik kunci yang berukuran sembarang. Ukuran kunci yang dapat diterima oleh blowfish adalah antara 32 *bit* hingga 448 *bit* menjadi beberapa *array* subkunci (*subkey*) dengan total 4168 *byte,* dengan ukuran

default sebesar 128 *bit*. Blowfish memanfaatkan teknik pemanipulasian *bit* dan teknik pemutaran ulang dan pergiliran kunci yang dilakukan sebanyak 16 kali. Kecepatan blowfish menyandi data adalah 32 *bits microprocessors* pada rentang dari 18 *clock cycles* per *byte*, bisa berjalan kurang dari 5k dari *memory* dan merupakan jenis penyandian yang mudah untuk diimplementasikan selain itu algoritma blowfish bebas paten dan akan berada pada *domain public*.

Bagian enkripsi dan dekripsi data terjadi dengan memanfaatkan perulangan 16 kali terhadap jaringan feistel. Setiap perulangan terdiri dari permutasi dengan masukan adalah kunci, dan substitusi data. Semua operasi dilakukan dengan memanfaatkan operator XOR dan penambahan. Operator penambahan dilakukan terhadap 4 *array lookup* yang dilakukan setiap putarannya.(Wikipedia, 2009)



Gambar 2.4 Algoritma Blowfish

Pada algoritma Blowfish, digunakan banyak *subkey*. Kunci-kunci ini harus dihitung atau dibangkitkan terlebih dahulu sebelum dilakukan enkripsi atau dekripsi data. Kunci- kunci yang digunakan antara lain terdiri dari, 18 buah 32-bit *subkey* yang tergabung dalam P-array (P1, P2, ..., P18). Selain itu, ada pula empat 32-bit S-box yang masing-masingnya memiliki 256 entri : S1,0, S1,1,..., S1,255; S2,0, S2,1,..,, S2,255; S3,0, S3,1,..., S3,255; S4,0, S4,1,..,, S4,255.

Untuk alur algoritma enkripsi dengan metoda Blowfish dijelaskan sebagai berikut: (ilmu komputer, 2009)

1. Bentuk inisial P *array* sebanyak 18 buah (P1,P2,.....P18) masing-masing bernilai 32 *bit*.

   Array P terdiri dari delapan belas kunci 32 *bit subkey* :

   $P_1,P_2,.....,P_{18}$

2. Bentuk Sbox (F) sebanyak 4 buah masing-masing bernilai 32 *bit* yang memiliki masukan 256.

   Empat 32 *bit* Sbox masing-masing mempunyai 256 *entri* :

   $S_{1,0},S_{1,1},.....,S_{1,255}$

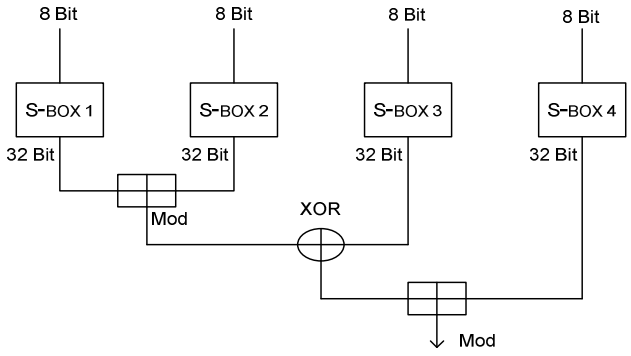
   $S_{2,0},S_{2,1},.....,S_{2,255}$

   $S_{3,0},S_{3,1},.....,S_{3,255}$

   $S_{4,0},S_{4,1},.....,S_{4,255}$

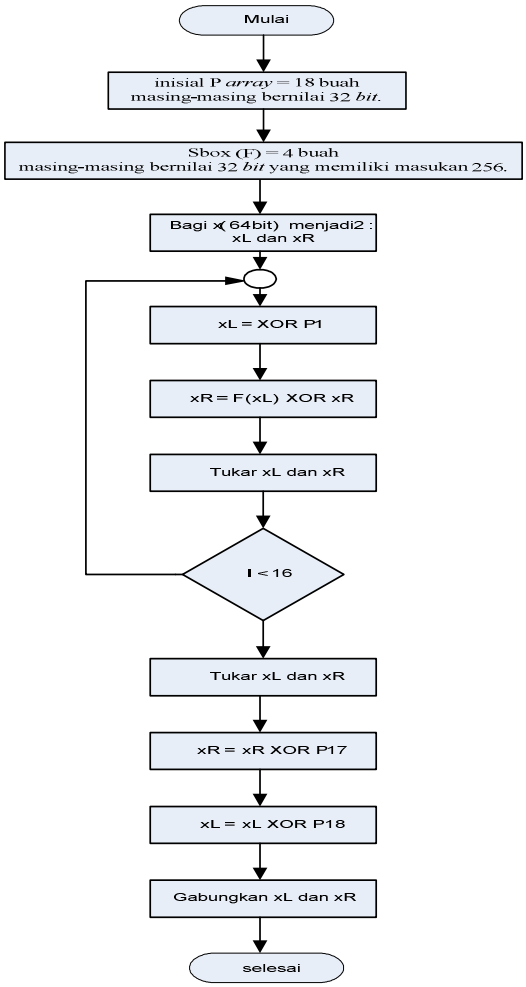
3. Plaintext yang akan dienkripsi diasumsikan sebagai masukan, Plaintext tersebut diambil sebanyak 64 *bit*, dan apabila kurang dari 64 *bit* maka kita tambahkan *bit*nya, supaya dalam operasi nanti sesuai dengan datanya.

4. Hasil pengambilan tadi dibagi 2, 32 *bit* pertama disebut XL, 32 *bit* yang kedua disebut XR.
5. Selanjutnya lakukan operasi XL = XL xor Pi dan XR = F(XL) xor XR
6. Hasil dari operasi diatas ditukar XL menjadi XR dan XR menjadi XL.
7. Lakukan sebanyak 16 kali, perulangan yang ke 16 lakukan lagi proses penukaran XL dan XR.
8. Pada proses ke 17 lakukan operasi untuk XR = XR xor P17 dan XL = XL xor P18.
9. Proses terakhir satukan kembali XL dan XR sehingga menjadi 64 *bit* kembali.

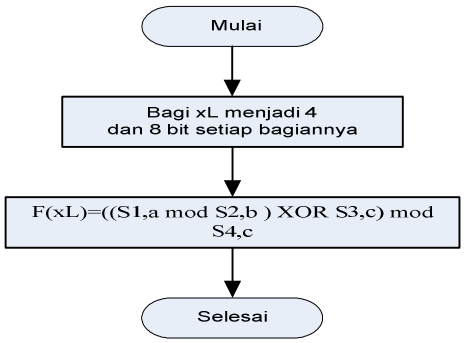


Gambar 2.5 Fungsi F Dalam Blowfish

Proses enkripsi algortima blowfish melalui beberapa tahap yang dapat dilihat pada gambar berikut:

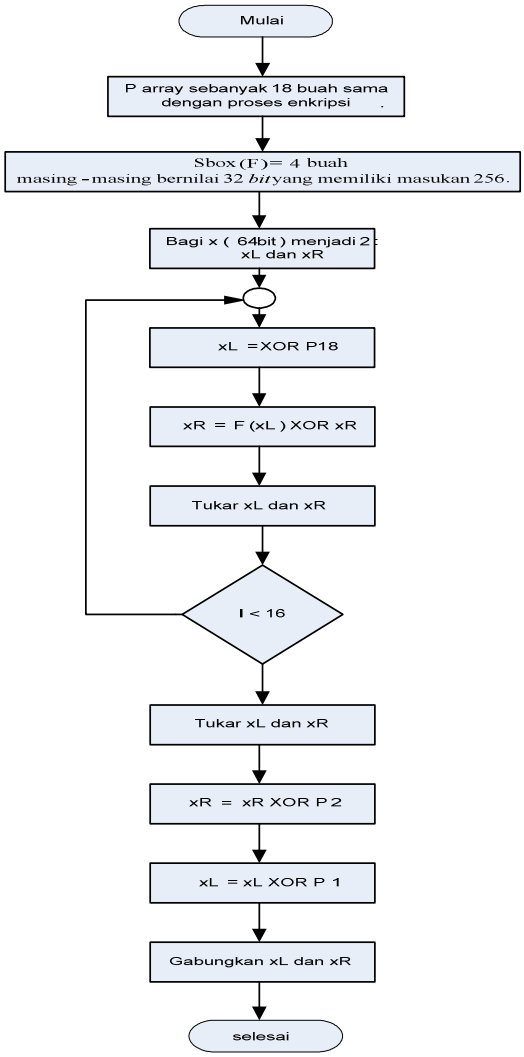


Gambar 2.6 Flowchart Enkripsi Algoritma Blowfish



Gambar 2.7 *Flowchart* Fungsi F (F (xL))

Proses dekripsi algoritma blowfish sama dengan proses enkripsi, kecuali P *array* (P1,P1,.....P18) digunakan dengan urutan terbalik atau di inverskan yang dapat dilihat pada gambar berikut :



Gambar 2.8 *Flowchart* Proses Dekripsi Algoritma Blowfish

2.3.1 Keamanan Blowfish

Keamanan dan kerahasiaan data sangat diperlukan baik dalam suatu organisasi maupun pribadi. Apalagi kalau data tersebut berada dalam suatu sistem komputer yang terhubung dalam jaringan komputer, bahkan dalam jaringan

internasional yang sering disebut internet. Kerahasiaan harus dilakukan dengan menjauhkan informasi dari orang-orang yang tidak berhak. Hal ini merupakan sesuatu yang harus diperhatikan ketika orang membahas keamanan data komputer. Keaslian berkaitan dengan siapa yang berbicara sebelum memberikan informasi yang sangat penting.

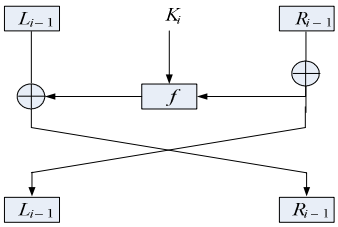
Tidak ada kelemahan yang berarti dari algoritma Blowfish yang dapat ditemukan sampai saat ini, kecuali adanya weak key, dimana dua entri dari S-Box mempunyai nilai yang sama. Tidak ada cara untuk mencek weak key sebelum melakukan key expantion. Bila dikhawatirkan, hal ini dapat mengurangi keamanannya sehingga dapat dibuat rutin untuk mengecek entri S-Box, walaupun hal ini tidak perlu.

Tidak ada kelemahan yang berarti dari algoritma Blowfish yang dapat ditemukan sampai saat ini bahkan belum ada Cryptanalisis yang berhasil memecahkan kelemahan algoritma Blowfish.(Andi, 2003)

2.3.2 Jaringan Feistel

Banyak algoritma menggunakan jaringan Feistel seperti Algoritma DES, Algoritma RC6, algoritma IDEA, algoritma Blowfish. Ide ini muncul pada tahun 1970. Bila diambil sebuah blok dengan panjang n kemudian blok tersebut dibagi dengan dua bagian sepanjang n/2 yaitu bagian L (kiri 0 dan bagian R (kanan, kemudian salah satu blok dikenakan fungsi f dan diputar. Demikian seterusnya selama beberapa kali putaran sehingga menghasilkan blok chiper yang lebih acak dan menghasilkan panjang yang genap. Definisikan sebuah chiper blok iterasi dimana output round ke-I ditentukan dari output round sebelumnya: (munir,2007).

K_i adalah subkey yang digunakan dalam putaran ke I dan f adalah fungsi *round arbitrary*



Gambar 2.9 Jaringan Feistel

2.4 Kriptanalis

Kriptografi menyebabkan timbulnya kriptanalis, yaitu ilmu pengetahuan dan seni untuk membongkar data acak. Praktisi dari kriptanalisis disebut kriptanalis. Kriptanalis berkembang sejalan dengan kriptografi. Setiap ada algoritma kriptografi baru yang dibuat oleh kriptografer, langsung diikuti oleh adanya upaya percobaan kriptanalis. Percobaan kriptanalis ini disebut attack (serangan).

Kriptanalisis mencoba megembalikan data jelas tanpa akses ke ke kunci kriptografi. Ukuran keberhasilan suatu upaya kriptanalisis adalah sampai sejauh mana keberhasilan diketahuinya data jelas atau kunci kriptografi.

Keseluruhan point dari kriptografi adalah menjaga kerahasiaan *plaintext* (atau kunci, atau keduanya) dari penyadap (*eavesdropper*) atau kriptanalis (*cryptanalyst*). (munir, 2007)

Nama lain penyadap:

1. Penyusup (*intruder*)
2. Penyerang (*attacker*)
3. Musuh (*enemy*, *adversaries*)

4. Pencegat (*interceptor*)

5. Lawan (*opponent*)

Penyadap berusaha mendapatkan data yang digunakan untuk kegiatan kriptanalisis (*cryptanalysis*).

Beberapa metode penyadapan data:

1. Wiretapping

Penyadap mencegat data yang ditransmisikan pada saluran kabel komunikasi dengan menggunakan sambungan perangkat keras.

2. Electromagnetic eavesdropping

Penyadap mencegat data yang ditransmisikan melalui saluran wireless, misalnya radio dan *microwive.*

3. *Acoustic Eavesdropping*.

Menangkap gelombang suara yang dihasilkan oleh suara manusia.

2.4.1 Jenis-jenis serangan:

Adapun jenis – jenis serangan untuk algoritma kunci simetris adalah sebagai berikut:(munir, 2007)

2.4.1.1 *Exhaustive attack* atau *brute force attack.*

Percobaan yang dibuat untuk mengungkap *plaintext* atau kunci dengan mencoba semua kemungkinan kunci (*trial* and *error).*

Asumsi yang digunakan:

1. Kriptanalis mengetahui algoritma kriptografi.
2. Kriptanalis memiliki sebagian *plaintext* dan *chipertext* yang bersesuaian.

Caranya: *plaintext* yang diketahui dienkripsikan dengan setiap kemungkinan kunci, dan hasilnya dibandingkan dengan *chiperteks* yang bersesuaian. Jika hanya *chiperteks* yang tersedia, *chiperteks* tersebut didekripsi dengan dengan setiap kemungkinan kunci dan plainteks hasilnya diperiksa apakah mengandung makna.

Misalkan sebuah sistem kriptografi membutuhkan kunci yang panjangnya 8 karakter, karakter dapat berupa angka (10 buah), huruf (26 huruf besar dan 26 huruf kecil), maka jumlah kunci yang harus dicoba adalah :

$$62 \times 62 \times 62 \times 62 \times 62 \times 62 \times 62 \times 62 = 62^8$$

Secara teori, serangan secara *exhaustive* ini dipastikan berhasil mengungkap plainteks tetapi dalam waktu yang sangat lama. Untuk menghadapi serangan ini, perancang kriptosistem (kriptografer) harus membuat kunci yang panjang dan tidak mudah ditebak.

2.4.1.2 *Analytical Attack*

Pada jenis serangan ini, kriptanalis tidak mencoba-coba semua kemungkinan kunci tetapi menganalisis kelemahan algoritma kriptografi untuk mengurangi kemungkinan kunci yang tidak mungkin ada.

Analisis dilakukan dengan dengan memecahkan persamaan-persamaan matematika yang mengandung peubah yang merepresentasikan plainteks atau kunci.Asumsi yang digunakan adalah kriptanalis mengetahui algoritma kriptografi. Untuk menghadapi serangan ini, kriptografer harus membuat algoritma kriptografi yang kompleks sedemikian sehingga plainteks merupakan fungsi matematika dari chiperteks dan kunci yang cukup kompleks, dan tiap kunci

merupakan fungsi matematika dari chiperteks dan plainteks yang cukup kompleks.

Metode *analytical attack* biasanya lebih cepat menemukan kunci dibandingkan dengan *exhaustive attack*.

1. Data yang digunakan untuk menyerang sistem kriptografi dapat dikategorikan sebagai berikut:
   1. ***Chipertext only*.**
   2. ***Known plaintext* dan *corresponding chipertext.***
   3. ***Chosen plaintext* dan *corresponding chipertext*.**
   4. ***Chosen chipertext* dan *corresponding plaintext*.**
2. Berdasarkan ketersediaan data yang ada, serangan terhadap sistem kriptografi dapat diklasifikasikan menjadi (asumsi yang digunakan: kriptanalis mengetahui algoritma kriptografi yang digunakan):
   1. ***Chipertext-only attack***

      Diberikan: $C_1 = E_k(P_1)$, $C_2 = E_k(P_2)$, …, $C_i = E_k(P_i)$

      Deduksi: P_1, P_2, …, P_i atau k untuk mendapatkan

      P_{i+1} dari $C_{i+1} = E_k(P_{i+1})$.

   2. ***Known-plaintext attack***

      Diberikan: P_1, $C_1 = E_k(P_1)$, P_2, $C_2 = E_k(P_2)$, …,

      P_i, $C_i = E_k(P_i)$

      Deduksi: k untuk mendapatkan P_{i+1} dari $C_{i+1} = E_k(P_{i+1})$.

3. Chosen-plaintext attack

Serangan jenis ini lebih hebat daripada *known-plaintext attack*, karena kriptanalis dapat memilih plainteks tertentu untuk dienkripsikan, yaitu plainteks-plainteks yang lebih mengarahkan penemuan kunci.

Diberikan: P_1, $C_1 = E_k(P_1)$, P_2, $C_2 = E_k(P_2)$, …,

P_i, $C_i = E_k(P_i)$ di mana kriptanalis dapat memilih diantara P_1, P_2, …, P_i

Deduksi: k untuk mendapatkan P_{i+1} dari $C_{i+1} = E_k(P_{i+1})$.

4. Adaptive-chosen-plaintext attack

Pada hal ini kriptanalis tidak hanya dapat memilih plaintext yang telah dienkripsi, tapi ia juga dapat memodifikasi pilihannya tersebut berdasarkan hasil enkripsi sebelumnya. Kriptanalis mengetahui *block* palaintext yang lebih kecil dan kemudian memilih yang lainnya berdasarkan hasil enkripsi pertama, kedua dan seterusnya.

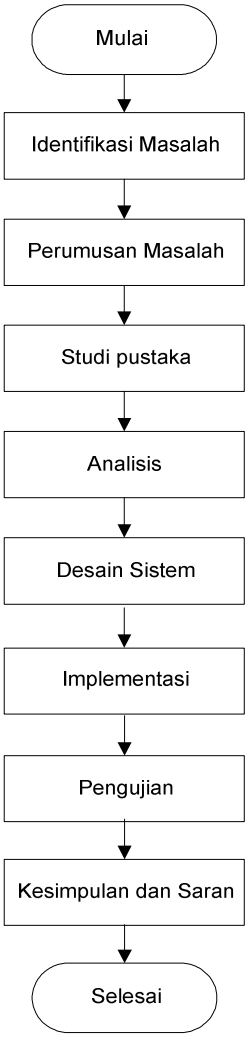
Sebuah algoritma kriptografi dikatakan aman (*computationally secure*) bila memenuhi tiga kriteria berikut:

1. Persamaan matematis yang menggambarkan operasi algoritma kriptografi sangat kompleks sehingga algoritma tidak mungkin dipecahkan secara analitik.
2. Biaya untuk memecahkan chiperteks melampaui nilai informasi yang terkandung di dalam chiperteks tersebut.
3. Waktu yang diperlukan untuk memecahkan chiperteks melampaui lamanya waktu informasi tersebut harus dijaga kerahasiaannya.

BAB III

METODOLOGI PENELITIAN

Pada bab ini akan dipaparkan tentang langkah-langkah yang digunakan untuk membahas permasalahan yang diambil dalam penelitian atau yang disebut dengan metodologi penelitian. Metodologi penelitian tugas akhir ini dapat digambarkan sebagai berikut :



Gambar 3.1 Tahapan Metodologi Penelitian

3.1 Identifikasi Masalah

Berdasarkan latar belakang yang telah dipaparkan sebelumnya pada bab I, maka permasalahan yang diangkat pada tugas akhir ini adalah bagaimana membangun aplikasi yang dapat mengamankan data *file*. Adapun permasalahan yang dapat diidentifikasi untuk pelaksanaan tugas akhir ini adalah :

1. meningkatnya para pengguna jaringan.
2. banyaknya kejahatan pada dunia internet.
3. kebutuhan pengiriman data berbentuk *file* dengan jarak jauh.
4. keamanan data yang sangat diperlukan dalam proses jaringan

3.2 Perumusan Masalah

Tujuan dari rumusan masalah adalah untuk mendapatkan point-point penting yang akan dibahas kemudian masalah tersebut akan diselesaikan satu persatu untuk mencapai tujuan akhir yang telah ditetapkan sebelumnya. Berdasarkan permasalahan yang telah diidentifikasi, maka dapat dirumuskan bahwa bagaimana membangun aplikasi yang dapat mengenkripsi dan dekripsi file dengan menggunakan algoritma blowfish

3.3 Studi Pustaka

Studi kepustakaan atau kajian pustaka dilakukan untuk mencari dan mempelajari serta mengumpulkan seluruh informasi yang terkait dan mendukung pelaksanaan penelitian pada tugas akhir ini. Studi kepustakaan atau kajian pustaka ini membahas tentang keamanan, enkripsi dan algoritma blowfish. Sumber kepustakaan diambil karya ilmiah yang berasal dari buku-buku maupun

internet. Karya ilmiah yang dimaksud adalah berupa tulisan ilmiah yang berbentuk artikel, prosiding, buku, *e-book* (buku elektronik), dan lain-lain.

3.4 Analisis

Tahap analisis kebutuhan aplikasi dilakukan untuk mengetahui kebutuhan pengguna terhadap aplikasi yang akan dikembangkan. Hal ini perlu dilakukan agar aplikasi yang dikembangkan sesuai dengan kebutuhan pengguna**.**

3.4.1 Analisis Kebutuhan Data

a. Data masukan (*input*)

   Data masukan yang dibutuhkan dalam pembuatan aplikasi ini adalah berupa :

   1. Data file yang diambil pada tempat penyimpanan.

b. Data proses

   Data proses yang berjalan pada aplikasi ini adalah :

   1 Proses enkripsi data, yang terdiri dari :

      a. Proses *input file*

      b. Proses *input* kunci (*password*)

   2 Proses dekripsi data, yang terdiri dari :

      a. Proses *input file* yang telah dienkripsi

      b. Proses *input* kata kunci yang sama.

c. Data keluaran (*output*)

   Data keluaran dari aplikasi ini berupa :

1. Hasil proses enkripsi *file* kedalam *format file* berbentuk ekstansi buatan oleh peneliti.
2. Hasil proses dekripsi *file* menjadi *file* asli sebelum dilakukan proses enkripsi.

3.4.2 Analisis Pengguna

Yang menjadi sasaran pengguna dari aplikasi ini adalah pihak pengguna internet yang ingin mendapatkan data dari internet. Jadi hanya satu jenis pengguna disini yaitu user biasa, tidak ada administrator atau pembeda antar pengguna.

3.5 Desain sistem

Pada tahapan ini dilakukan perancangan antarmuka (*input* dan *output*), perancangan proses dan merancang prosedur atau algoritma berdasarkan hasil dari analisa kebutuhan aplikasi yang telah dilakukan sebelumnya. Aplikasi yang dibuat tidak memerlukan media *database*. Penyimpanan hasil adalah pada komputer local (*Hard disk*). Pada tahap desain ini, *tool* yang digunakan adalah *Microsoft Office Visio* 2003.

3.6 Implementasi

Hasil rancangan yang telah dilakukan pada tahap desain kemudian diubah menjadi bentuk yang dimengerti oleh mesin menggunakan bahasa pemrograman agar bisa diimplementasikan oleh pengguna.

3.7 Pengujian

Sebelum perangkat lunak dapat digunakan, maka harus dilakukan pengujian terlebih dahulu. Pengujian difokuskan pada pencarian semua kemungkinan kesalahan, dan memeriksa apakah sistem telah sesuai dengan yang diinginkan serta memperbaiki kesalahan yang terdapat pada sistem.

Pengujian pada penelitian tugas akhir ini dilakukan terhadap 4 hal, yaitu :

1. Pengujian fungsi-fungsi aplikasi.

   Pengujian ini bertujuan untuk menentukan apakah semua fungsi pada aplikasi berjalan dengan baik. Menguji keberhasilan proses enkripsi dan dekripsi terhadap semua *file input*. dengan cara menentukan file biner terhadap file yang akan di uji dan melakukan proses sesuai dengan algoritma blowfish yang diterapkan. Pengujian dilakukan dengan memeriksa semua fungsi yang terdapat pada aplikasi.

2. Pengujian modul.

   Pengujian modul ini merupakan hasil pengujian implementasi aplikasi secara detail mengenai item-item yang terdapat pada setiap tampilan proses. Pengujian modul meliputi pengujian modul enkripsi file dan dekripsi file.

3. Pengujian terhadap waktu dengan jenis dan ukuran file yang berbeda.

   Pengujian terhadap waktu dengan jenis dan ukuran file yang berbeda bertujuan untuk melihat waktu terhadap jenis dan ukuran file dan memberikan *list* hasil dari pengujian. Sehingga dari *list* tersebut dapat

ditentukan berapa waktu yang digunakan untuk memperoleh hasil enkripsi dan dekripsi terhadap jenis dan ukuran file yang berbeda.

4. Pengujian terhadap *attack*.

   Pengujian terhadap *attack* bertujuan memberikan list pengujian dan membuktikan kekuatan algoritma blowfish dari serangan dengan menggunakan jenis serangan (*attack*) *Exhaustive attack* atau *brute force attack.* Percobaan yang dibuat untuk mengungkap *plaintext* atau kunci dengan mencoba semua kemungkinan kunci (*trial and error*).

3.8 Kesimpulan dan saran

Setelah permasalahan dapat diselesaikan dan kemudian memberi ringkasan tentang segala sesuatu yang telah diuraikan pada bab-bab sebelumnya dan harus menghubungkan setiap kelompok data dengan permasalahan untuk sampai pada kesimpulan tertentu secara teoritik maupun praktis. Sedangkan saran berisi alternatif yang diajukan peneliti agar permasalahan yang ada dapat dipecahkan sebaik-baiknya di waktu mendatang.

BAB IV

ANALISIS DAN PERANCANGAN

4.1 Analisis

Analisis perangkat lunak dalam membangun aplikasi kriptografi pengamanan data pada file dengan algoritma blowfish meliputi analisis masalah, analisis metode, analisis kebutuhan sistem dari aplikasi yang akan dibuat, sehingga aplikasi yang dibangun sesuai dengan maksud dan tujuan yang ingin dicapai dalam penelitian ini.

4.1.1 Analisis Masalah

Keamanan data dan informasi merupakan hal yang sangat penting dalam sistem jaringan komputer. Semakin berkembangnya teknologi informasi, maka berkembang pula teknik kejahatan yang berupa perusakan maupun pencurian data oleh pihak yang tidak memiliki wewenang atas data tersebut. Ada beberapa bentuk penyerangan terhadap data dan informasi, seperti *hacker, cracker, trojan force attack,* dan lain-lain. Oleh karena itu, pada saat ini telah dilakukan berbagai upaya untuk menjaga keamanan data dan mengatasi serangan-serangan tersebut.

Salah satu cara dalam mengimplementasikan keamanan data dan informasi dalam sistem jaringan komputer adalah dengan teknik kriptografi. Kriptografi merupakan salah satu cara untuk mengamankan *file* atau data rahasia pada suatu media yang tampak agar tidak dapat dibuka, kecuali bagi orang yang mengerti kuncinya. Sebelumnya telah dilakukan teknik kriptografi pada *file*

dengan batasan tertentu, namun *file* yang perlu diamankan pada saat ini tidak hanya *file* tersebut. Oleh karena itu dilakukan pengembangan dari *file* tersebut yang berupa penerapan teknik kriptografi pada semua *file*. Algoritma yang digunakan dalam kriptografi *file* ini adalah blowfish. Blowfish memanfaatkan teknik pemanipulasian *bit* dan teknik pemutaran ulang dan pergiliran kunci yang dilakukan sebanyak 16 kali.

4.1.2 Analisis Metode

Didalam proses ini terdapat *file* biner hasil konvers file asli kedalam biner dan kunci yang dibangkitkan terlebih dahulu sebelum dilakukan proses enkripsi atau dekripsi data. Kunci- kunci yang digunakan antara lain terdiri dari, 18 buah yang memiliki 32-bit *subkey* yang tergabung dalam P-array (P1, P2, ..., P18). Selain itu, ada empat 32-bit S-box yang masing-masingnya memiliki 256 entri : S1,0, S1,1, ...,S1,255. S2,0, S2,1, ..,S2,255. S3,0, S3,1, ...,S3,255. S4,0, S4,1, .., S4,255.

Konsep dari algortima blowfish adalah bahwa setiap bit *file* akan ditambahkan dengan bit yang berasal dari kunci – kunci pada P-array dan S-box. Dalam penambahan bit harus mencukupi 64 bit, jika melebihi dari 64 bit maka akan dilakukan proses perulangan. Berikut ini gambaran dari proses tersebut.

File Awal (contoh simulasi file biner hasil konvers dari file asli)

11000011 11010011 11000011 10100110

File Kriptografi (contoh simulasi file biner hasil konvers file asli dengan kunci blowfish)

11100111 1100001 10001010 01010100 10101010 10100101 10101010 11001100

4.1.3 Analisis Kebutuhan Sistem

Setelah melakukan analisis, maka dapat diketahui kebutuhan sistem yang akan dibangun, seperti kebutuhan masukan sistem, kebutuhan keluaran sistem, kebutuhan fungsi dan antar muka sistem yang akan dibuat, serta kebutuhan perangkat lunak yang dibutuhkan dalam pembuatan aplikasi, sehingga sistem yang dibangun sesuai dengan yang diharapkan.

4.1.3.1 Kebutuhan Masukan

Data masukan yang dibutuhkan dalam pembuatan aplikasi ini adalah berupa :

1. *Input file* data yang akan di-kriptografi.
2. Kata kunci yang digunakan untuk keamanan kriptografi dan dibutuhkan untuk membuka proses enkripsi.

4.1.3.2 Kebutuhan Fungsi atau Proses

Proses yang berjalan pada aplikasi ini adalah :

1. Proses pemilihan *file*.

   Merupakan proses pemilihan *file* sebagai masalah yang akan diproses.

2. Proses konversi data *file* ke biner.

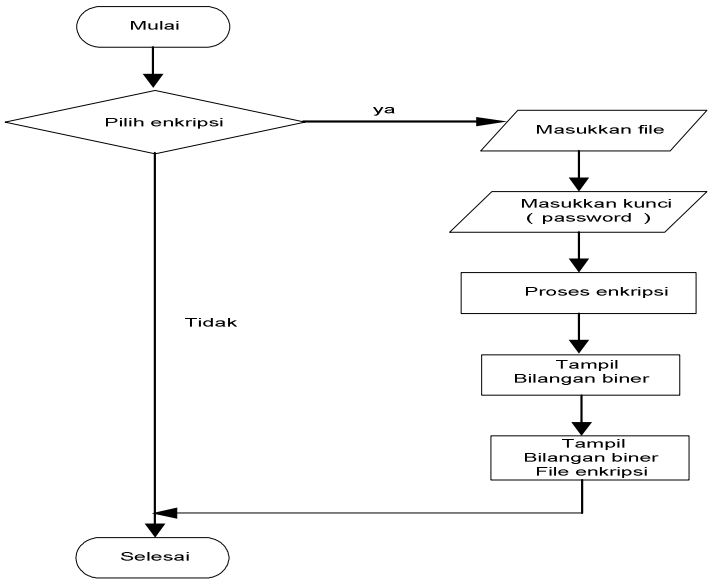
   file harus dikonversi ke dalam bentuk biner. Dalam penelitian ini, proses konversi *file* menggunakan aplikasi tambahan sebagai pendukung pada tugas akhir. Setelah mendapatkan file biner kemudian file tersebut diproses sesuai dengan algoitma blowfish sebagai pengamanan data.

3. Proses *input* kata kunci (*password*)

   Proses *input* kata kunci (*password*) untuk membuka *file* yang telah di enkripsi. Kata kunci yang digunakan merupakan kata kunci yang di-*input* kan pada saat proses enkripsi dan dekripsi *file*.

4. Proses enkripsi file

   Proses enkripsi file berisi tombol enkripsi, informasi tentang ukuran file yang di masukkan.proses enkripsi file dimulai dari inisialisasi masukan aplikasi kemudian pemberian kunci dan menyimpan file enkripsi tersebut kedalam file berekstensi *.blw. Proses enkripsi melalui beberapa tahap yang dapat dilihat pada gambar berikut:

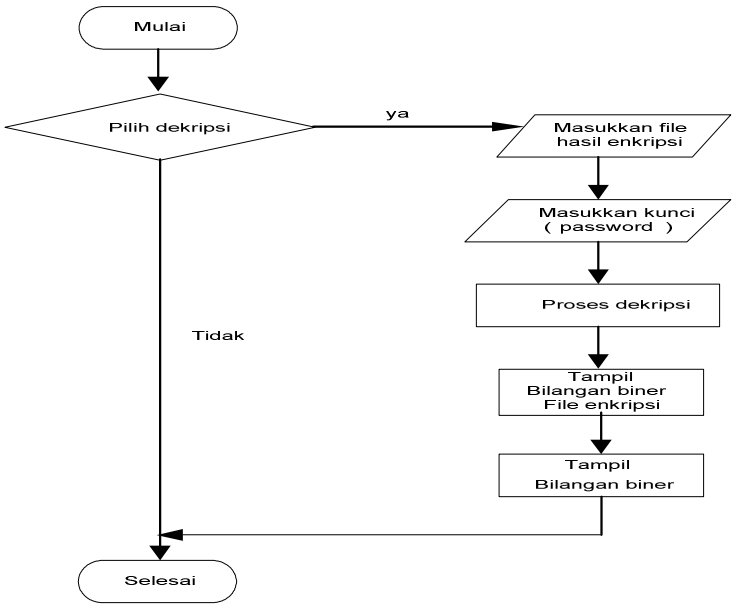


Gambar 4.1 Flowchart Enkripsi File

5. Proses dekripsi file

   Proses dekripsi file membuka kembali file yang telah dienkripsi kemudian memasukkan kunci untuk membuka file enkripsi tersebut.

Proses dekripsi melalui beberapa tahap yang dapat dilihat pada gambar berikut:



Gambar 4.2 Flowchart Dekripsi File

6. Proses enkripsi algortima blowfish

   Proses enkripsi algortima blowfish merupakan chiper blok 64 bit dan melakukan proses iterasi sebanyak 16 putaran.

7. Proses dekripsi algoritma blowfish

   Proses dekripsi algoritma blowfish merupakan chiper blok 64 bit dan melakukan proses iterasi sebanyak 16 putaran.

Berikut ini merupakan contoh dari proses enkripsi algoritma blowfish pada file yang bertipe *.text dengan bit yang diperoleh **10110101 10101010 01011010 10101011 01110111 11110111 01010101 10000100** adalah sebagai berikut :

1. Inisialkan P *array* 18 buah

   P1 = **10010101 01110101 01010111 01010111**.

   Array P terdiri dari 18 kunci 32 *bit subkey* : P_1,P_2,.....,P_{18}

2. *File* yang telah dikonversi kedalam biner diambil sebanyak 64 *bit*, dan apabila kurang dari 64 *bit* maka kita tambahkan *bit*nya, supaya dalam operasi nanti sesuai dengan datanya, kemudian dibagi menjadi 2 dari bit **10110101 10101010 01011010 10101011 01110111 11110111 01010101 10000100,**

   xL = **10110101 10101010 01011010 10101011**

   xR = **01110111 11110111 01010101 10000100**

3. Lakukan operasi xL = xL XOR P1

   xL = **10110101 10101010 01011010 10101011** XOR **10010101 01110101 01010111 01010111**

   xL = **00100000 11011111 00001101 11111100**

4. Lakukan operasi xR = F(xL) XOR xR, masuk kedalam proses pada fungsi F

   Lakukan operasi fungsi F, xL dibagi menjadi 4 buah (a,b,c,d) masing - masing bernilai 8 bit yang mempunyai 256 entri dan setiap entri memiliki 32 bit ($S_{1,0}$,$S_{1,1}$,.....,$S_{1,255}$).

   Ubah 8 bit kedalam bentuk bilangan desimal, dari hasil yang diperoleh ambil bit yang terdapat pada 256 entri (masing-masing entri memiliki 32 bit) sesuai dengan jumlah desimal pada hasil pencarian.

   a. 00100000 = 32_{10} ,

      Diperoleh bit pada S1,32 = **10101001 10101000 10101000 10101010**

b. 11011111 = 223_{10}

Diperoleh bit pada S1,223 = **10001011 10101000 10101001 10101001**

c. 00001101 = 13_{10}

Diperoleh bit pada S1,13 = **10100010 10010100 10101010 00101010**

d. 11111100 = 252_{10}

Diperoleh bit pada S1,252 = **10101000 00101011 10000010 10101001**

S1,a ubah kedalam biner dari setiap 8 bit kemudian bagi dengan S1,b yang telah diubah kedalam biner dari setiap 8 bit, begitu juga pada proses s1,d

F(xL) = ((S1,a mod S2,b)XOR S3,c) mod S4,d

= ((**10101001 10101000 10101000 10101010 mod 10001011 10101000 10101001 10101001**) XOR S3,c) mod S4,d

= (**00011101 11111111 11111111 00000001** XOR **10100010 10010100 10101010 00101010)** mod S4,d

= **10100101 11010100 01111101 10101000** mod **10101000 00101011 10000010 10101001**

= **01010010 11101010 01111101 10101000**

Didapat untuk bit pada fungsi f yang pertama **01010010 11101010 01111101 10101000**

5. Lakukan proses f(xL) XOR xR

xR = **01010010 11101010 01111101 10101000** XOR **01110111 11110111 01010101 10000100**

= 00100101 00011101 00101000 00101100

6. Tukar xL dan xR,

   xL = **00100101 00011101 00101000 00101100**

   xR = **00100000 11011111 00001101 11111100**

7. Lakukan proses tesebut sebanyak 16 kali putaran dengan proses yang sama.

8. Setelah proses perulangan sebanyak 16 kali selesai,

   Didapat untuk xL = **10110100 01001010 01100010 01111101** dan xR = **10001011 00111101 01011110 01111100** kemudian lakukan proses,

   xR = xR XOR P17

   xL = xL XOR P18

   dari inisial P *array* didapat untuk P17 dan P 18 bitnya

   P17 = **01110101 10010101 01101010 01010110**

   P18 = **11101010 01010010 00111100 01111100**

   Maka,

   xR = **10001011 00111101 01011110 01111100** XOR **01110101 10010101 01101010 01010110 = 11111110 10101000 00110100 00101010** dan

   xL = **10110100 01001010 01100010 01111101** XOR **11101010 01010010 00111100 01111100 = 01011110 00011000 01011110 00000001**

9. Lakukan proses penggabungan antara xL dan xR = **01011110 00011000 01011110 00000001 11111110 10101000 00110100 00101010**

10. Proses enkripsi telah berhasil, dengan hasil bit **01011110 00011000 01011110 00000001 11111110 10101000 00110100 00101010**

Berikut ini merupakan contoh dari proses dekripsi algoritma blowfish pada file yang telah dienkripsi dengan bit yang diperoleh **01011110 00011000 01011110 00000001 11111110 10101000 00110100 00101010** adalah sebagai berikut :

11. P *array* 18 buah sama dengan *array* yang ada pada proses enkripsi

    P18 = **11101010 01010010 00111100 01111100**.

12. *File* yang telah dikonversi kedalam biner diambil sebanyak 64 *bit*, dan apabila kurang dari 64 *bit* maka kita tambahkan *bit*nya, supaya dalam operasi nanti sesuai dengan datanya, kemudian dibagi menjadi 2 dari bit **01011110 00011000 01011110 00000001 11111110 10101000 00110100 00101010,**

    xL = **01011110 00011000 01011110 00000001**

    xR = **11111110 10101000 00110100 00101010**

13. Lakukan operasi xL = xL XOR P18

    xL = **01011110 00011000 01011110 00000001** XOR **11101010 01010010 00111100 01111100**

    xL = **10110100 01001010 01100010 01111101**

14. Lakukan operasi xR = F(xL) XOR xR, masuk kedalam proses pada fungsi F

    Lakukan operasi fungsi F, xL dibagi menjadi 4 buah (a,b,c,d) masing-masing bernilai 8 bit yang mempunyai 256 entri dan setiap entri memiliki 32 bit ($S_{1,0}$,$S_{1,1}$,.....,$S_{1,255}$).

    Ubah 8 bit kedalam bentuk bilangan desimal, dari hasil yang diperoleh ambil bit yang terdapat pada 256 entri (masing-masing entri memiliki 32 bit) sesuai dengan jumlah desimal pada hasil pencarian.

a. 10110100 = 180_{10} ,

Diperoleh bit pada S1,180 = **11001111 10101000 00101001 10101000**

b. 01001010 = 74_{10}

Diperoleh bit pada S2,74 = **10101001 10001011 11101111 0101000**

c. 01100010 = 82_{10}

Diperoleh bit pada S3,82 = **00010001 11101110 10101010 10101010**

d. 01111101 = 125_{10}

Diperoleh bit pada S4,125 = **11100000 00101011 10101001 00011100**

S1,a ubah kedalam biner dari setiap 8 bit kemudian bagi dengan S1,b yang telah diubah kedalam biner dari setiap 8 bit, begitu juga pada proses s1,d

F(xL) = ((S1,a mod S2,b)XOR S3,c) mod S4,d

= ((**11001111 10101000 00101001 10101000** mod **10101001 10001011 11101111 0101000**) XOR S3,c) mod S4,d

= (**00100110 00011100 00111010 01011000** XOR **00010001 11101110 10101010 10101010)** mod S4,d

= **10001110 00110111 10111000 11110001** mod **11100000 00101011 10101001 00011100**

= **10001110 00110111 10111000 11110001**

Didapat untuk bit pada fungsi f yang pertama **10001110 00110111 10111000 11110001**

15. Lakukan proses f(xL) XOR xR

xR = **10001110 00110111 10111000 11110001** XOR **11111110 10101000 00110100 00101010**

= 01110000 10011111 10001100 11011000

Tukar xL dan xR,

xL = **01110000 10011111 10001100 11011000**

xR = **01011110 00011000 01011110 00000001**

16. Lakukan proses tesebut sebanyak 16 kali putaran dengan proses yang sama.
17. Setelah proses perulangan sebanyak 16 kali selesai,

    Didapat untuk xL = **00110101 10101110 01011101 00101100** dan xR = **10001011 00111101 01011110 01111100** kemudian lakukan proses,

    xR = xR XOR P2

    xL = xL XOR P1

    dari inisial P *array* didapat untuk P2 dan P 1 bitnya

    P2 = **01110101 10010101 01101010 01010110**

    P1 = **10010101 01110101 01010111 01010111**

    Maka,

    xR = **01011001 10001110 11000010 00011011** XOR **00101110 01111001 00010111 10011111 = 01110111 11110111 01010101 10000100** dan

    xL = **00110101 10101110 01011101 00101100** XOR **10010101 01110101 01010111 01010111 = 01011110 00011000 01011110 00000001**
18. Lakukan proses penggabungan antara xL dan xR = **10110101 10101010 01011010 10101011 01110111 11110111 01010101 10000100**

19. Proses dekripsi telah berhasil, dengan hasil bit sama dengan bit pada file asli,

10110101 10101010 01011010 10101011 01110111 11110111 01010101 10000100

4.1.3.3 Analisis Keluaran Data

Setelah proses kriptografi dilakukan, aplikasi ini akan menampilkan sebuah keluaran (*output*) yang sesuai dengan masukan (*input*) pada saat kriptografi dilakukan, keluaran dari aplikasi kriptografi ini berupa file asli yang sebelumnya di lakukan proses pemutaran pada *file* yang berbentuk biner.

4.2 Perancangan

Perancangan perangkat lunak dalam membangun aplikasi kriptografi *file* sebagai keamanan data dengan algoritma blowfish meliputi perancangan antarmuka proses enkripsi dan perancangan antarmuka proses dekripsi.

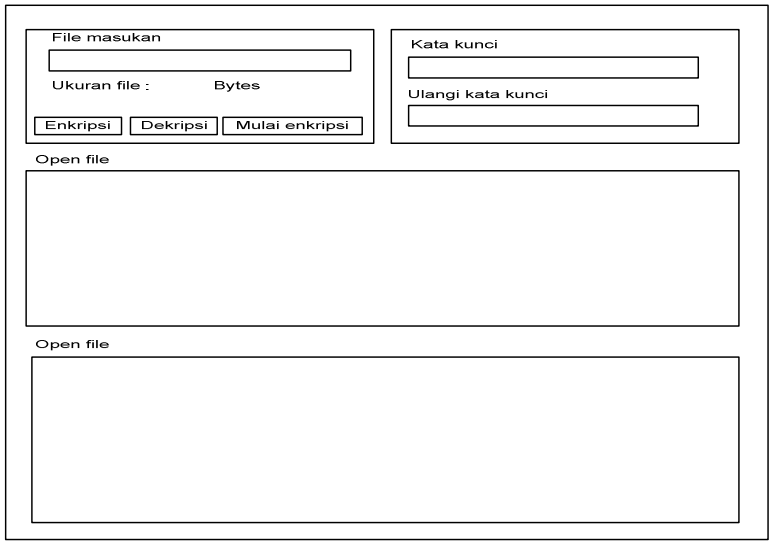
4.2.1 Perancangan Antarmuka Aplikasi Yang Akan Dibangun

Antarmuka atau *interface* merupakan suatu sarana yang memungkinkan terjadinya interaksi antara manusia dan komputer. Oleh sebab itu, *interface* dari sebuah perangkat lunak yang akan dibangun harus bersifat *user friendly* yang bertujuan agar pengguna (*user*) dapat mengerti dengan mudah dan memahami cara menggunakan perangkat lunak ini.

4.2.1.1 Perancangan Antarmuka Proses Enkripsi

Gambaran antarmuka (*interface*) yang dibutuhkan aplikasi yang akan dibangun ini terdiri dari beberapa bagian yang diakses *user*. Salah satu

perancangan tampilan enkripsi pada kriptografi ini adalah sebagai berikut pada gambar 4.3 :



Gambar 4.3 Antarmuka Menu Enkripsi

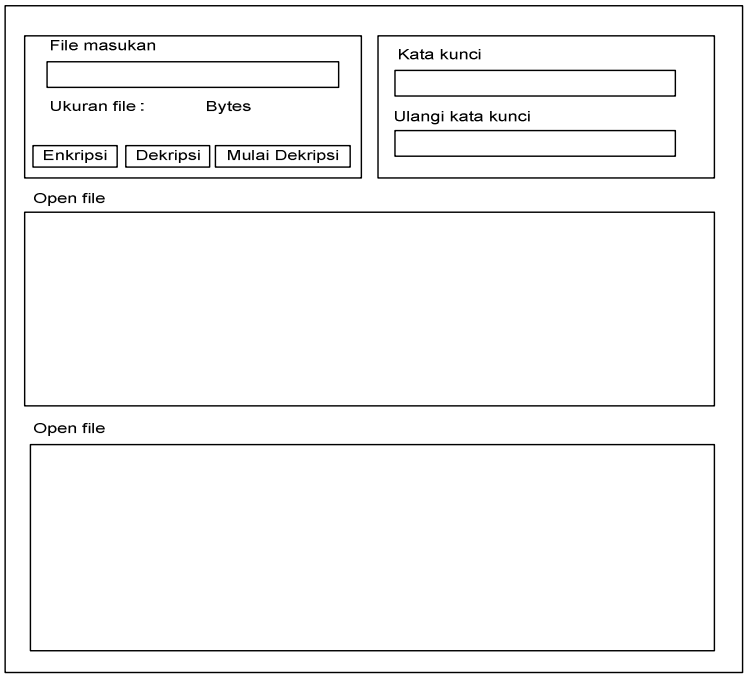
Keterangan antarmuka dapat dilihat pada tabel berikut:

Tabel 4.1 Tabel Keterangan Antarmuka Menu Enkripsi

Objek	Properti	Pengaturan
Label 1	Caption	File masukan
Label 2	Caption	Ukuran file
Label 3	Caption	Kata kunci
Label 4	Caption	Ulangi kata kunci
Label 5	Caption	Menampilkan file biner asal
Label 6	Caption	Menampilkan file biner enkripsi
button 1	Caption	Enkripsi
button 2	Caption	Dekripsi
button 3	Caption	Mulai enkripsi
Textbox 1	Font	MS Sans Serif
Textbox 2	Font	MS Sans Serif
Textbox 3	Font	MS Sans Serif
Textbox 4	Font	MS Sans Serif

4.2.1.2 Perancangan Antarmuka Proses Dekripsi

Perancangan antarmuka proses dekripsi meliputi beberapa proses yang harus dijalankan oleh *user*. Berikut ini merupakan antarmuka menu untuk melakukan dekripsi file hasil kriptografi. Untuk melakukan proses dekripsi *user* harus menekan pilihan *button* "Dekripsi" yang terdapat pada menu (gambar 4.4). tampilan sebagai berikut:



Gambar 4.4 Antarmuka Menu Dekripsi

Keterangan antarmuka dapat dilihat pada tabel berikut:

Tabel 4.2 Tabel Keterangan Antarmuka Menu Dekripsi

Objek	Properti	Pengaturan
Label 1	Caption	File masukan
Label 2	Caption	Ukuran file
Label 3	Caption	Kata kunci
Label 4	Caption	Ulangi kata kunci
Label 5	Caption	Open file bit enkripsi

Label 6	Caption	Open file bit dekripsi
button 1	Caption	Enkripsi
button 2	Caption	Dekripsi
button 3	Caption	Mulai dekripsi
Textbox 1	Font	MS Sans Serif
Textbox 2	Font	MS Sans Serif
Textbox 3	Font	MS Sans Serif
Textbox 4	Font	MS Sans Serif

BAB V

IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi Sistem

Implementasi merupakan tahapan dimana sistem sudah bisa dioperasikan. Hal ini dilakukan setelah penulisan kode program. Pada tahap implementasi sistem ini, diharapkan sistem yang telah dirancang siap untuk dioperasikan pada keadaan yang sebenarnya, sehingga akan diketahui apakah sistem yang dibuat benar-benar sesuai seperti yang diharapkan. Implementasi perangkat lunak ini dibuat dan dibangun dengan bantuan bahasa pemrograman *Borland Delphi 7.0*.

5.1.1 Alasan Pemilihan Perangkat Lunak

Pemilihan perangkat lunak ini didasarkan pertimbangan sebagai berikut:

1. *borland delphi 7.0* merupakan *event-driven programming* (pemograman terkendali kejadian) artinya program menunggu sampai adanya respon dari pemakai berupa *event* atau kejadian tertentu (tombol diklik, menu dipilih, dan lain-lain). Ketika *event* terdeteksi, kode yang berhubungan dengan *event* (prosedur *event*) akan dijalankan.
2. *borland delphi 7.0* mempunyai kemampuan untuk memanfaatkan *windows* dan tidak memerlukan pemrograman khusus untuk menampilkan jendela (*window*), dan cara penggunaannya berbasis *visual* seperti aplikasi *windows* lainnya.

3. *borland delphi 7.0* dapat digunakan untuk menguji program (*debugging*) dan menghasilkan program akhir EXE, yang bersifat *executable* atau dapat langsung dijalankan.
4. *borland delhpi 7.0* sering dieksploitasi untuk menunjang efesiensi bisnis karena kemudahan mendesain *user interface*.

5.1.2 Batasan Implementasi

Batasan implementasi pada penulisan tugas akhir ini adalah :

1. Perangkat lunak tidak memberikan solusi keamanan pada penghapusan terhadap *file* tersebut.
2. Dekripsi file tidak dapat secara langsung menunjukan file asli jika terjadi proses enkripsi secara berulang kali pada file yang sama.

5.1.3 Lingkungan Implementasi

Implementasi dilakukan pada lingkungan perangkat keras dan lingkungan perangkat lunak.

1. Perangkat Keras

   Perangkat keras yang digunakan mempunyai spesifikasi sebagai berikut:

   a. *Processor* Intel Core 2 Duo 1,83 GHz
   b. *Memory* 2 GB
   c. *Harddisk* berkapasitas 120 GB
   d. Monitor
   e. *Speaker*
   f. *Keyboard*
   g. *Mouse*

2. Perangkat Lunak

Perangkat lunak dalam implementasi ini menggunakan:

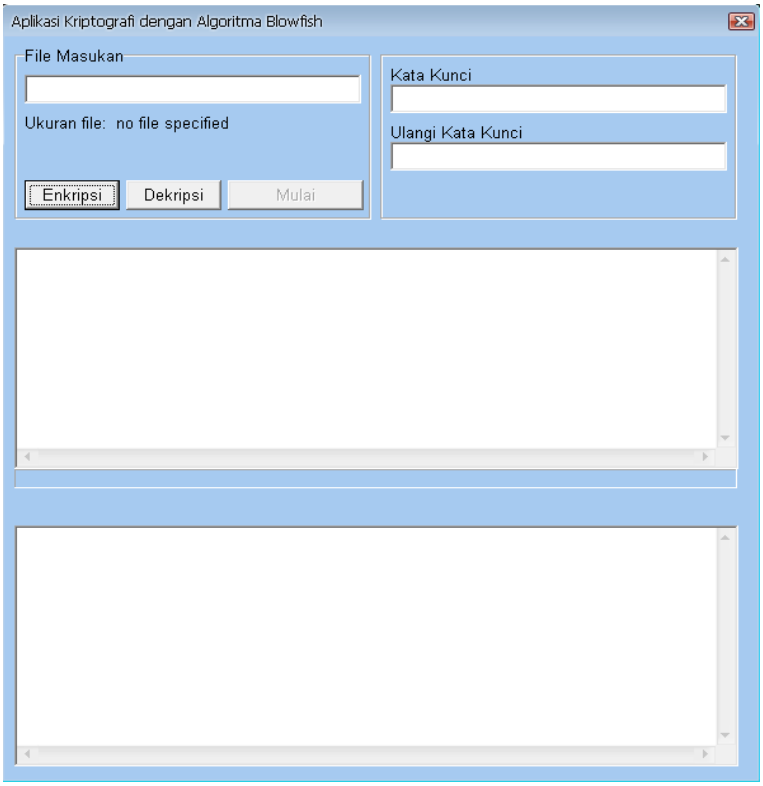
a. Sistem Operasi *Windows XP Professional*.

b. Bahasa pemrograman *Borland Delphi 7.0*.

5.1.4 Tampilan aplikasi

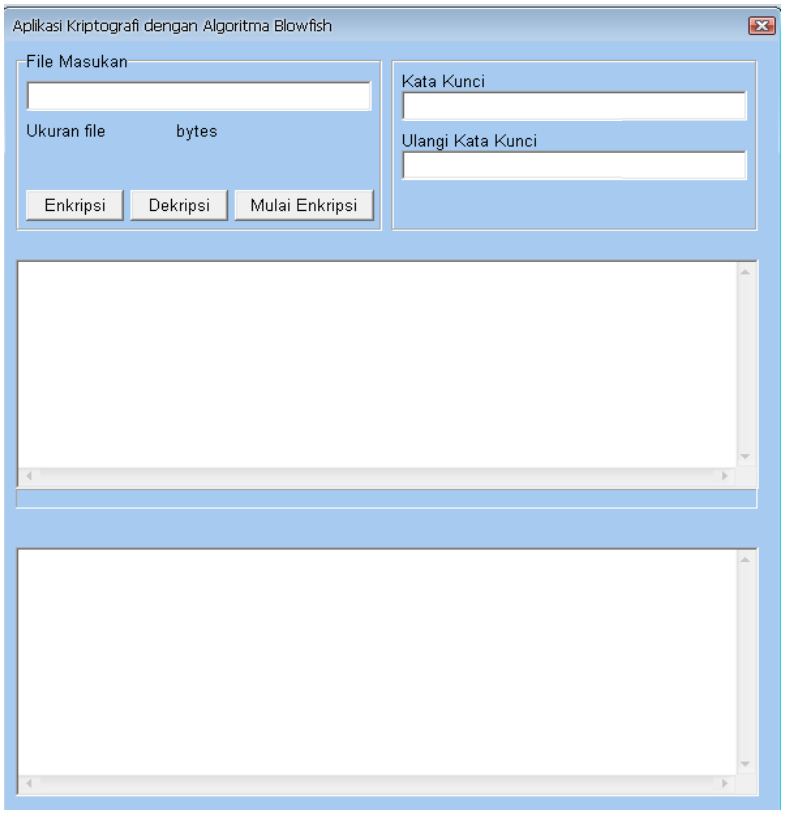
Aplikasi ini dirancang khusus untuk membantu dalam mengamankan data penting dengan teknik kriptografi. Aplikasi keamanan data menggunakan teknik kriptografi pada file ini secara umum diperlihatkan melalui tampilan menu utama aplikasi seperti pada Gambar 5.1.

Aplikasi ini terdiri dari beberapa menu yang mempunyai aturan tertentu yang harus dijalankan secara berurutan.



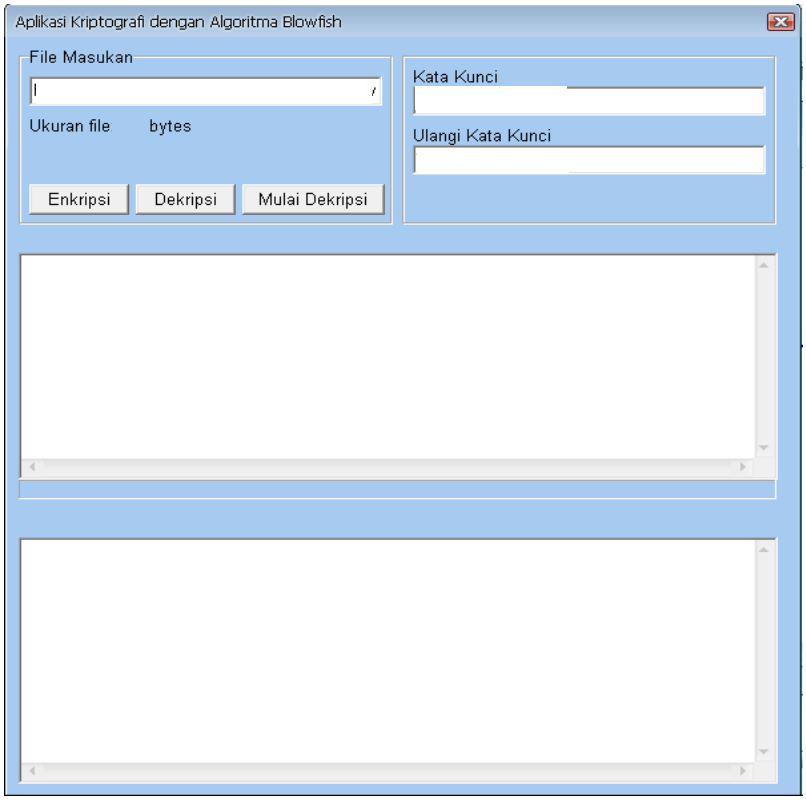
Gambar 5.1 Tampilan Menu Utama Kriptografi Pada File

Menu berikutnya merupakan menu enkripsi yang terdiri dari sumber file dan kunci (*password*). Pada *menu* ini juga dapat menampilkan proses bilangan biner file asli dengan file yang telah di enkripsi (gambar 5.2).



Gambar 5.2 Tampilan Menu Enkripsi Kriptografi Pada File

Menu berikutnya merupakan menu dekripsi yang terdiri dari sumber file yang telah di enkripsi dan kunci (*password*) yang sama pada saat enkripsi. Pada *menu* ini juga dapat menampilkan proses bilangan biner file yang telah di enkripsi dengan file asli (gambar 5.3).



Gambar 5.3 Tampilan Menu Dekripsi Kriptografi Pada File

5.2 Pengujian Sistem

Setelah tahap implementasi dilakukan maka dilanjutkan dengan melakukan pengujian keberhasilan terhadap aplikasi yang telah dibuat. Tahap pengujian diperlukan untuk mengetahui apakah aplikasi telah siap untuk digunakan oleh pengguna. Pengujian dilakukan dengan memperlihatkan proses bilangan biner baik sumber file asli maupun file yang telah dilakukan kriptografi. Kemudian pembuktian dilakukan untuk membuktikan bahwa file yang dihasilkan setelah melakukan proses pemutaran adalah sesuai dengan masukan (*input*) yang hanya menggunakan 64 bit setiap prosesnya.

5.2.1 Lingkungan Pengujian Sistem

Pengujian sistem ini dilakukan pada lingkungan perangkat lunak dan perangkat keras sesuai dengan lingkungan implementasi.

5.2.2 Rencana Pengujian

Pengujian pada penelitian tugas akhir ini difokuskan terhadap 4 hal, yaitu:

1. Pengujian fungsi-fungsi aplikasi.

   Pengujian ini bertujuan untuk menentukan apakah semua fungsi pada aplikasi berjalan dengan baik, yaitu dengan menguji keberhasilan proses enkripsi dan dekripsi terhadap semua *file inpu*, dengan cara menentukan file biner terhadap file yang akan di uji dan melakukan proses sesuai dengan algoritma blowfish yang diterapkan. Pengujian dilakukan dengan memeriksa semua fungsi yang terdapat pada aplikasi.

2. Pengujian modul.

   Pengujian modul ini merupakan hasil pengujian implementasi aplikasi secara detail mengenai item-item yang terdapat pada setiap tampilan proses. Pengujian modul meliputi pengujian modul enkripsi file dan dekripsi file.

3. Pengujian terhadap waktu dengan jenis dan ukuran file yang berbeda.

   Pengujian terhadap waktu dengan jenis dan ukuran file yang berbeda bertujuan untuk melihat waktu terhadap jenis dan ukuran file dan memberikan *list* hasil dari pengujian. Sehingga dari *list* tersebut

dapat ditentukan berapa waktu yang digunakan untuk memperoleh hasil enkripsi dan dekripsi terhadap jenis dan ukuran file yang berbeda.

4. Pengujian terhadap *attack*.

   Pengujian terhadap *attack* bertujuan memberikan list pengujian dan membuktikan kekuatan algoritma blowfish dari serangan dengan menggunakan jenis serangan (*attack*) *Exhaustive attack* atau *brute force attack.* Percobaan yang dibuat untuk mengungkap *plaintext* atau kunci dengan mencoba semua kemungkinan kunci (*trial and error*).

Pengujian keberhasilan aplikasi pada tugas akhir ini melakukan proses menampilkan file asli dan file yang telah di enkripsi. Hal ini dilakukan dengan tujuan untuk mengetahui perubahan atau penambahan pada biner *file* data yang dilakukan oleh sistem pada masing-masing kondisi.

5.2.3 Pengujian Tampilan Aplikasi

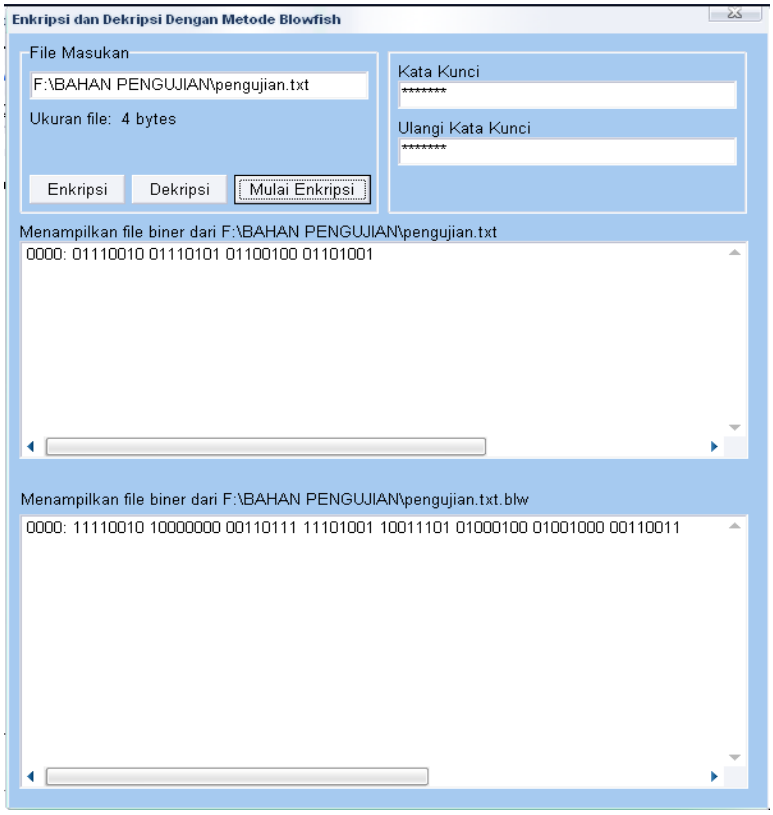
Pengujian dilakukan dengan melihat tampilan masing-masing menu yang ada dalam setiap urutan proses enkripsi maupun pada proses dekripsi

5.2.3.1 Pengujian Tampilan Proses Enkripsi

Pada tampilan awal diperlihatkan *menu* pilihan yang berfungsi untuk menentukan proses yang dilakukan yaitu proses enkripsi dan proses dekripsi yang ditunjukkan oleh gambar 5.1. Untuk melakukan proses enkripsi pilih tombol **”enkripsi”**, ambil file yang akan diproses sebagai kasus peneliti mengambil file

dalam format *.txt **”pengujian*.txt”** dan memasukkan kata kunci **”ujicoba”** sebagai pembuka pada saat proses dekripsi

Proses kriptografi dengan algoritma blowfish merupakan proses pemutaran bilangan biner. Untuk mejalankan aplikasi kriptografi dilakukan dengan mengklik tombol ”**mulai enkripsi**”, dan dilanjutkan dengan menampilkan bilangan biner pada file tersebut dan file yang telah dienkripsi. Tampilan *menu* proses dapat dilihat pada gambar berikut:

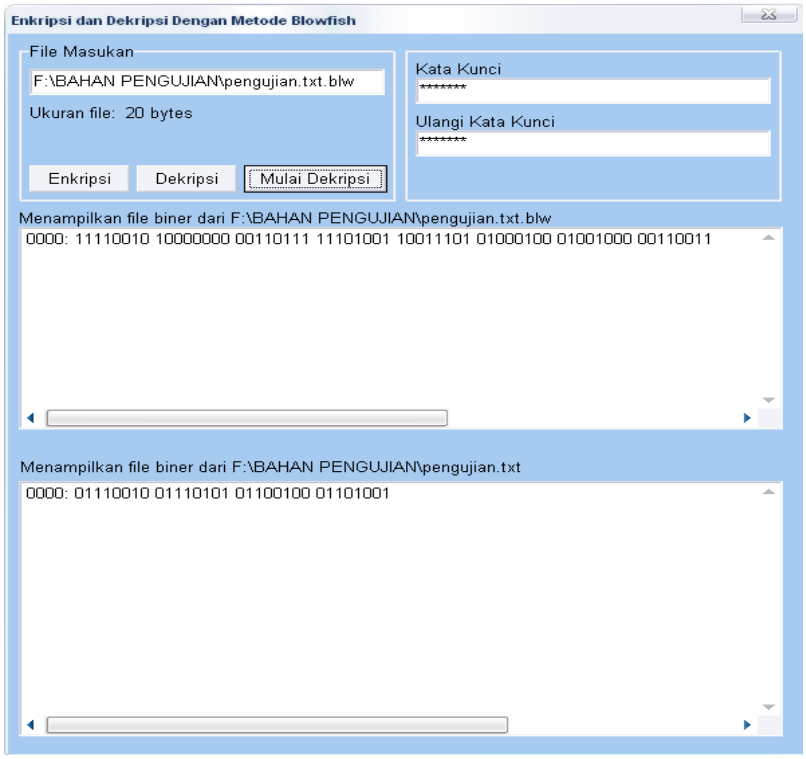


5.4 Tampilan Proses Enkripsi

5.2.3.2 Pengujian Tampilan Proses Dekripsi

Pada tampilan awal diperlihatkan *menu* pilihan yang berfungsi untuk menentukan proses yang dilakukan yaitu proses enkripsi dan proses dekripsi yang ditunjukkan oleh gambar 5.1. Untuk melakukan proses dekripsi pilih tombol **"dekripsi"**, ambil file yang akan diproses sebagai kasus peneliti mengambil file dalam format *.blw **"pengujian*.txt.blw"** dan memasukan kata kunci yang sama pada saat proses enkripsi.

Proses kriptografi dengan algoritma blowfish merupakan proses pemutaran pada bilangan biner. Untuk mejalankan aplikasi kriptografi dilakukan dengan mengklik tombol "**mulai dekripsi**", dan dilanjutkan dengan menampilkan bilangan biner pada file yang telah dienkripsi dan file asli. Tampilan *menu* proses dapat dilihat pada gambar berikut:



5.5 Tampilan Proses Dekripsi

5.2.3.3 Pengujian Tampilan Hasil Proses Enkripsi

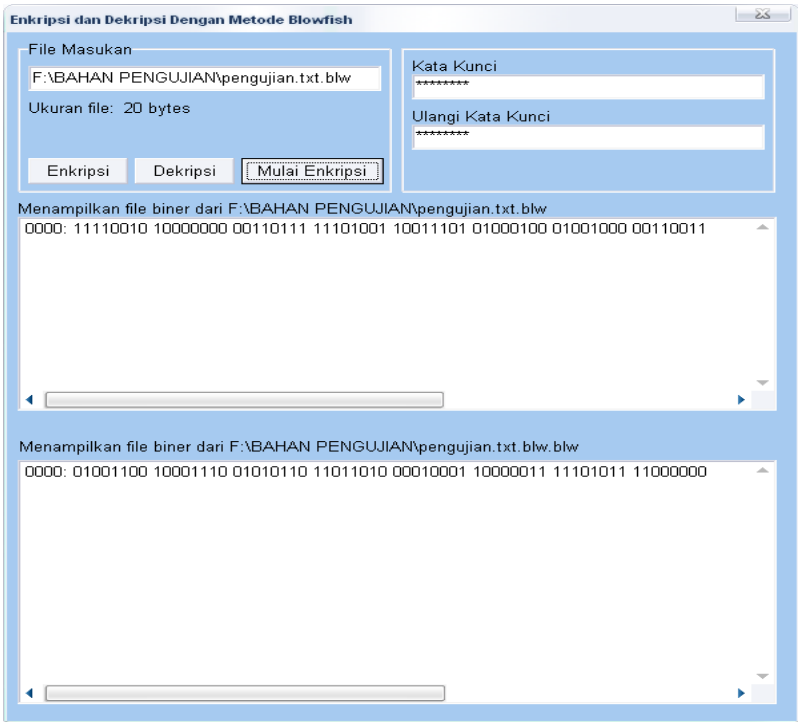
Menampilkan file data dengan ekstansi buatan penulis yang sebelumnya disisipkan ke dalam file berupa kunci, kata kunci(*password*). Dari hasil pengujian diperoleh *type file* yang berbeda dari file awal.

5.6 Tampilan Hasil Proses Enkripsi

5.2.3.4 Pengujian Tampilan Proses Enkripsi Yang Telah Di Enkripsi

Untuk melakukan proses enkripsi berulang pilih tombol **”enkripsi”**, ambil file yang akan diproses sebagai kasus peneliti mengambil file dalam format *.txt.blw **”pengujian*.txt.blw”** dan memasukkan kata kunci **”ujicoba2”** sebagai pembuka pada saat proses dekripsi

Tampilan *menu* proses dapat dilihat pada gambar berikut:



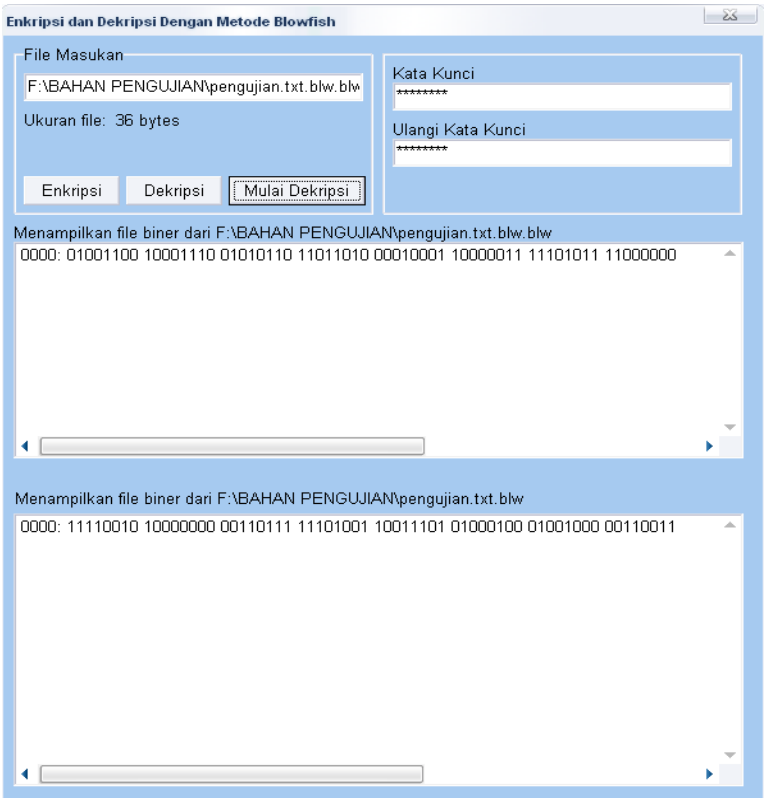
5.7 Tampilan Hasil Enkripsi Kedua

5.2.3.5 Pengujian Tampilan Proses Dekripsi Yang Telah Di Enkripsi Dua Kali

Untuk melakukan proses dekripsi yang telah di enkripsi berulang pilih tombol **"dekripsi"**, ambil file yang telah dienkripsi sebanyak yang telah diproses untuk dapat menunjukan file asli. Proses kriptografi dengan algoritma blowfish merupakan proses pemutaran bilangan biner.

5.2.3.5.1 Pengujian Tampilan Proses Dekripsi Pertama

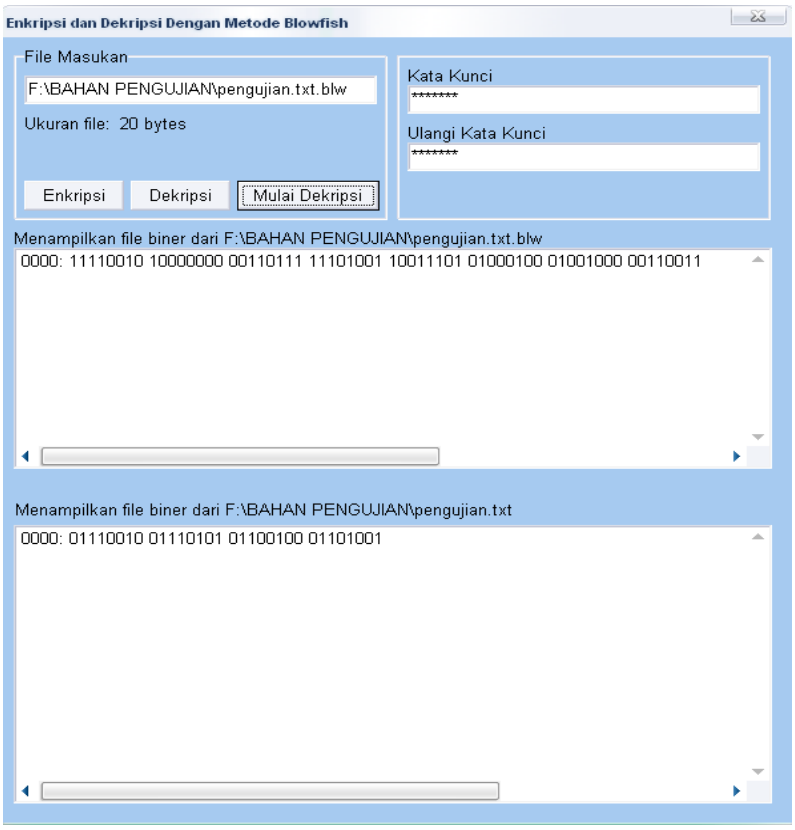
Untuk mejalankan aplikasi kriptografi pada dekripsi dilakukan dengan mengklik tombol "**mulai dekripsi**", dan dilanjutkan dengan menampilkan bilangan biner pada file yang telah dienkripsi dua kali dan file enkripsi awal. Tampilan *menu* proses dapat dilihat pada gambar berikut:



5.8 Tampilan Hasil Dekripsi Pertama

5.2.3.5.2 Pengujian Tampilan Proses Dekripsi Kedua

Untuk mejalankan aplikasi kriptografi dilakukan dengan mengklik tombol ”**mulai dekripsi**”, dan dilanjutkan dengan menampilkan bilangan biner pada file file enkripsi awal dan file asli. Tampilan *menu* proses dapat dilihat pada gambar berikut:



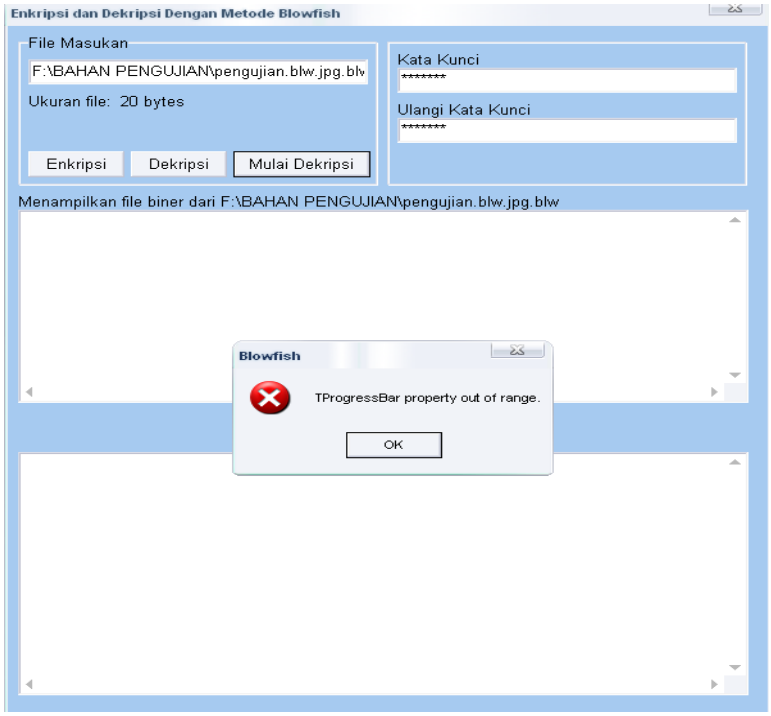
5.9 Tampilan Hasil Dekripsi Kedua

5.2.3.6 Pengujian Tampilan Proses Dekripsi Jika Format Berubah

Pada tampilan awal diperlihatkan *menu* pilihan yang berfungsi untuk menentukan proses yang dilakukan yaitu proses enkripsi dan proses dekripsi yang ditunjukkan oleh gambar 5.1. Untuk melakukan proses dekripsi pilih tombol **”dekripsi”**, ambil file yang akan diproses sebagai kasus peneliti mengambil file

yang telah dirubah menjadi format *.blw.jpg.blw **”pengujian*.blw.jpg.blw”** dan memasukkan kata kunci yang sama pada saat proses enkripsi.

Tampilan *menu* proses dapat dilihat pada gambar berikut:



6.0 Tampilan Proses Dekripsi Jika Format Berubah

5.2.3.7 Pengujian Tampilan Pesan Proses Enkripsi Dan Dekripsi

Tampilan pesan berikut ini akan muncul apabila pada menu enkripsi dan dekripsi akan diproses setelah kunci (*password*) dimasukkan dan aplikasi akan mengeluarkan kotak pesan sebagai berikut.

6.1 Tampilan Proses Memulai Enkripsi

6.2 Tampilan Memulai Proses Dekripsi

5.2.3.8 Pengujian Tampilan Pesan Hasil Proses

Setelah seluruh proses selesai dijalankan, maka aplikasi akan memberikan pesan aplikasi berhasil dilakukan. Pesan aplikasi dapat dilihat pada gambar berikut ini:

6.3 Tampilan Proses Aplikasi Berhasil

5.2.4 Pengujian Modul

Pengujian modul ini merupakan hasil pengujian implementasi aplikasi secara detail mengenai item-item yang terdapat pada setiap tampilan proses enkripsi file. Pengujian modul meliputi pengujian modul enkripsi file dan dekripsi file.

5.2.4.1 Pengujian Modul Enkripsi File

Merupakan hasil pengujian implementasi aplikasi secara detail mengenai item-item yang terdapat pada setiap tampilan proses enkripsi file.

5.2.4.1.1 Pengujian Tahap I Mencari sumber *File*

Prekondisi

1. Sudah ada sumber file didalam perangkat penyimpanan yang akan dilakukan pengujian

Tabel 5.1 Tabel Butir Uji Pengujian Tahap 1 Mencari Sumber *File*

Deskripsi	Prekondisi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian tahap 1 mencari sumber *file*	Tampilan layar menu utama, ada dua fasislitas option, Enkripsi atau dekripsi	1. pada sumber file tekan tombol ”Enkripsi” 2. cari file yang akan diuji	Sumber file, besar file	Proses pembacaan file berhasil, tidak ada instruksi error	Proses pembacaan file berhasil, tidak ada instruksi error	Proses pembacaan file berhasil, tidak ada instruksi error	Di terima

5.2.4.1.2 Pengujian Tahap 2 Memasukkan Kata Kunci.

Prekondisi

1. Tahap 1 telah dilalui tidak ada instruksi error

Tabel 5.2 Tabel Butir Uji Pengujian Tahap 2 Memasukkan Kata Kunci

Deskripsi	Prekondisi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian Tahap 2 Memasukkan kata kunci	Tahap 1 telah dilalui tidak ada instruksi error	1. tampil menu memasukkan kata kunci 2. ulangi kata kunci yang sama	Kata kunci sebagai enkripsi dan dekripsi file	Kata kunci berhasil diproses, tidak ada instruksi error	Kata kunci berhasil diproses, tidak ada instruksi error	Kata kunci berhasil diproses, tidak ada instruksi error	Di terima

5.2.4.1.3 Pengujian Tahap 3 Proses Enkripsi File.

Prekondisi

1. Tahap 1 dan 2 telah dilalui tidak ada instruksi error

Tabel 5.3 Tabel Butir Uji Pengujian Tahap 3 Proses Enkripsi File

Des Kripsi	Prekondisi	Prosedur Pengujian	Masu Kan	Keluaran yang Di harapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian proses 3 proses enkripsi file	Tahap 1 dan 2 telah dilalui tidak ada instruksi error	1. tampil menu proses mulai enkripsi 2. Klik tombol ”ok” Setelah proses berhasil akan tampil pesan ”proses enkripsi telah selesai lanjutkan dengan membuka file biner” 3. klik tombol ”ok” untuk menampilkan bilangan biner file asli dengan file hasil enkripsi	File biner asli dengan file biner hasil enkripsi	Proses konversi file biner berhasil, dalam proses ini tidak ada instruksi error	Proses konversi file biner berhasil, dalam proses ini tidak ada instruksi error	Proses konversi file biner berhasil, dalam proses ini tidak ada instruksi error	Di terima

5.2.4.2 Pengujian Modul Dekripsi File.

Pengujian modul ini merupakan hasil pengujian implementasi aplikasi secara detail mengenai item-item yang terdapat pada setiap tampilan proses dekripsi file.

5.2.4.2.1 Pengujian Tahap 1 Menentukan File Hasil kriptografi.

Prekondisi

1. Sudah ada sumber file hasil kriptografi didalam perangkat penyimpanan yang akan dilakukan pengujian

Tabel 5.4 Tabel Butir Uji Pengujian Menentukan File Hasil Kriptografi Yang Akan Didekripsi

Deskripsi	Prekondisi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian menentukan *file* hasil kriptografi yang akan didekripsi	Sudah ada sumber file hasil kriptografi	1. Pada menu tekan tombol ”Dekripsi” 2. Cari file hasil kriptografi	*File* hasil kriptografi	Tampil Proses Pembacaan file yang telah dienkripsi dan tidak ada instruksi error	Tampil Proses Pembacaan file yang telah dienkripsi dan tidak ada instruksi error	Tampil Proses Pembacaan file yang telah dienkripsi dan tidak ada instruksi error	Di terima

5.2.4.2.2 Pengujian Tahap 2 Memasukkan Kata Kunci.

. Prekondisi

1. Tahap 1 telah dilalui tidak ada instruksi error

Tabel 5.5 Tabel Butir Uji Pengujian Tahap 2 Memasukkan Kata Kunci

Deskripsi	Prekondisi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian Tahap 2 Memasukkan Kata Kunci	Tahap 1 dan 2 telah dilalui tidak ada instruksi error	1. tampil menu untuk memasukkan kata kunci 2. Silahkan ketik kata kuncinya sesuai kata kunci pada saat enkripsi 3. ulangi kata kunci yang sama	kata kunci	Jika kata kunci benar masuk keproses berikutnya jika salah hasil proses dekripsi tidak menemukan file asli dan tidak ada instruksi error.	Jika kata kunci benar masuk keproses berikutnya jika salah hasil proses dekripsi tidak menemukan file asli dan tidak ada instruksi error.	Jika kata kunci benar masuk keproses berikutnya jika salah hasil proses dekripsi tidak menemukan file asli dan tidak ada instruksi error.	Di terima

5.2.4.2.3 Pengujian Tahap 3 Proses Dekripsi File.

Prekondisi

1. Tahap 1 dan 2 telah dilalui tidak ada instruksi error

Tabel 5.6 Tabel Butir Uji Pengujian Tahap 3 Proses Dekripsi File

Des Kripsi	Prekon disi	Prosedur Pengujian	Masu Kan	Keluaran yang Di harapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesim pulan
Pengujia n tahap 3 proses dekripsi file	Tahap 1 dan 2 telah dilalui tidak ada instruksi error	1. tampil menu proses mulai Dekripsi 2. Klik tombol ”ok” Setelah proses berhasil akan tampil pesan ”proses dekripsi telah selesai lanjutkan dengan membuka file biner” 3. klik tombol ”ok” untuk menampilkan bilangan biner file hasil enkripsi dengan file asli	File biner hasil enkripsi dengan File biner asli	Proses konversi file biner berhasil, dalam proses ini tidak ada instruksi error	Proses konversi file biner berhasil, dalam proses ini tidak ada instruksi error	Proses konversi file biner berhasil, dalam proses ini tidak ada instruksi error	Di terima

5.2.5 Pengujian Terhadap Waktu Dengan Jenis Dan Ukuran File Yang Berbeda.

Pengujian terhadap semua file dan waktu yang dibutuhkan untuk proses enkripsi dan dekripsi dilakukan dengan melihat waktu untuk membuktikan dan memberikan *list* hasil dari pengujian. Sehingga dari *list* tersebut dapat ditentukan berapa waktu yang digunakan untuk memperoleh hasil enkripsi dan dekripsi terhadap besarnya ukuran file.

Pengujian dilakukan dengan melakukan percobaan terhadap file antara lain *.Text. *.Doc, *.Pdf, *.ppt, *. Jpg, *.Gif, * Mp3, *.Mpeg, *.Avi, *.Rar.

Tabel 5.7 Pengujian terhadap *file* *.text

No	Nama File	Ukuran File (Bytes)	Kata kunci (Password)	Waktu Enkripsi	Waktu Dekripsi
1	Pengujian1.text	3,878 bytes	123qwe	01.34 s	01.34 s
2	Pengujian2.text	24,242 bytes	123qwe	04.57 s	04.57 s
3	Pengujian3.text	78,597 bytes	123qwe	16.77 s	16.77 s
4	Pengujian4.text	185,911 bytes	123qwe	29.75 s	29.75 s
5	Pengujian5.text	478,530 bytes	123qwe	02:03.33 s	02:03.33 s

Tabel diatas merupakan pengujian terhadap waktu dengan jenis dan ukuran file *.text yang berbeda menggunakan algoritma blowfish dengan panjang kunci 32 bit dan jumlah iterasi 16 putaran.

Tabel 5.8 Pengujian terhadap *file* *.doc

No	Nama File	Ukuran File (Bytes)	Kata kunci (Password)	Waktu Enkripsi	Waktu Dekripsi
1	Pengujian1.doc	61,440 bytes	123456	14.03 s	14.03 s
2	Pengujian2.doc	136,192 bytes	123456	20.37 s	20.37 s
3	Pengujian3.doc	161,792 bytes	123456	23.58 s	23.58 s
4	Pengujian4.doc	367,164 bytes	123456	01:46.16 s	01:46.16 s
5	Pengujian5.doc	1,559,040 bytes	123456	05:34.35 s	05:34.35 s

Tabel diatas merupakan pengujian terhadap waktu dengan jenis dan ukuran file *.doc yang berbeda menggunakan algoritma blowfish dengan panjang kunci 32 bit dan jumlah iterasi 16 putaran.

Tabel 5.9 Pengujian terhadap *file* *.pdf

No	Nama File	Ukuran File (Bytes)	Kata kunci (Password)	Waktu Enkripsi	Waktu Dekripsi
1	Pengujian1.pdf	115,190 bytes	654321	16.80 s	16.80 s
2	Pengujian2. pdf	212,351 bytes	654321	26.13 s	26.13 s
3	Pengujian3. pdf	348,841 bytes	654321	45.67 s	45.76 s
4	Pengujian4. pdf	537,012 bytes	654321	01:12.80 s	01:12.80 s
5	Pengujian5. pdf	661,961 bytes	654321	01:29.79 s	01:29.79 s

Tabel diatas merupakan pengujian terhadap waktu dengan jenis dan ukuran file *.pdf yang berbeda menggunakan algoritma blowfish dengan panjang kunci 32 bit dan jumlah iterasi 16 putaran.

Tabel 6.0 Pengujian terhadap *file* *.ppt

No	Nama File	Ukuran File (Bytes)	Kata kunci (Password)	Waktu Enkripsi	Waktu Dekripsi
1	Pengujian1.ppt	45.544 bytes	qwerty	08.40 s	08.40 s
2	Pengujian2. ppt	90,112 bytes	qwerty	14.75 s	14.75 s
3	Pengujian3. ppt	252,928 bytes	qwerty	32.63 s	32.63 s
4	Pengujian4. ppt	390,114 bytes	qwerty	50.49 s	50.49 s
5	Pengujian5. ppt	1,536,512 bytes	qwerty	05:30.28 s	05:30.28 s

Tabel diatas merupakan pengujian terhadap waktu dengan jenis dan ukuran file *.ppt yang berbeda menggunakan algoritma blowfish dengan panjang kunci 32 bit dan jumlah iterasi 16 putaran.

Tabel 6.1 Pengujian terhadap *file* *.Jpg

No	Nama File	Ukuran File (Bytes)	Kata kunci (Password)	Waktu Enkripsi	Waktu Dekripsi
1	Pengujian1.Jpg	37,112 bytes	qweasd	08.93 s	08.93 s
2	Pengujian2. Jpg	229,803 bytes	qweasd	35.40 s	35.40 s
3	Pengujian3. Jpg	538,056 bytes	qweasd	01:21.70 s	01:21.70 s
4	Pengujian4. Jpg	1,304,709 bytes	qweasd	04:41.06 s	04:41.06 s
5	Pengujian5. Jpg	3,145,373 bytes	qweasd	09:33.78 s	09:33.78 s

Tabel diatas merupakan pengujian terhadap waktu dengan jenis dan ukuran file *.jpg yang berbeda menggunakan algoritma blowfish dengan panjang kunci 32 bit dan jumlah iterasi 16 putaran.

Tabel 6.2 Pengujian terhadap *file* *.gif

No	Nama File	Ukuran File (Bytes)	Kata kunci (Password)	Waktu Enkripsi	Waktu Dekripsi
1	Pengujian1. gif	93,174 bytes	asdqwe	16.38 s	16.38 s
2	Pengujian2.gif	130,765 bytes	asdqwe	24.48 s	24.48 s
3	Pengujian3. gif	150,231 bytes	asdqwe	26.82 s	26.82 s
4	Pengujian4. gif	157,713 bytes	asdqwe	30.66 s	30.66 s
5	Pengujian5. gif	209,645 bytes	asdqwe	40.11 s	40.11 s

Tabel diatas merupakan pengujian terhadap waktu dengan jenis dan ukuran file *.gif yang berbeda menggunakan algoritma blowfish dengan panjang kunci 32 bit dan jumlah iterasi 16 putaran.

Tabel 6.3 Pengujian terhadap *file* *.mp3

No	Nama File	Ukuran File (Bytes)	Kata kunci (Password)	Waktu Enkripsi	Waktu Dekripsi
1	Pengujian1.mp3	291,065 bytes	098765	54.22 s	54.22 s
2	Pengujian2. mp3	388,545 bytes	098765	90.56 s	90.56 s
3	Pengujian3. mp3	598,517 bytes	098765	01:90.67 s	01:90.67 s
4	Pengujian4. mp3	949,562 bytes	098765	03:90.12 s	04:30.12 s
5	Pengujian5. mp3	1,032,387 bytes	098765	01:01.23 s	01:01.23 s

Tabel diatas merupakan pengujian terhadap waktu dengan jenis dan ukuran file *.mp3 yang berbeda menggunakan algoritma blowfish dengan panjang kunci 32 bit dan jumlah iterasi 16 putaran.

Tabel 6.4 Pengujian terhadap *file* *.mpeg

No	Nama File	Ukuran File (Bytes)	Kata kunci (Password)	Waktu Enkripsi	Waktu Dekripsi
1	Pengujian1.mpeg	368,076 bytes	567890	90.88 s	90.88 s
2	Pengujian2. mpeg	380,960 bytes	654321	95.86 s	95.86 s
3	Pengujian3. mpeg	585,732 bytes	654321	01:20.56 s	01:20.56 s
4	Pengujian4. mpeg	681,968 bytes	654321	01:56.23 s	01:56.23 s
5	Pengujian5. mpeg	1,435,676 bytes	654321	04:15.45 s	04:15.45 s

Tabel diatas merupakan pengujian terhadap waktu dengan jenis dan ukuran file *.mpeg yang berbeda menggunakan algoritma blowfish dengan panjang kunci 32 bit dan jumlah iterasi 16 putaran.

Tabel 6.5 Pengujian terhadap *file* *.avi

No	Nama File	Ukuran File (Bytes)	Kata kunci (Password)	Waktu Enkripsi	Waktu Dekripsi
1	Pengujian1.avi	593,402 bytes	zxcvbn	01:50.21 s	01:50.21 s
2	Pengujian2. avi	768,216 bytes	zxcvbn	02:20.13 s	02:20.13 s
3	Pengujian3. avi	834,816 bytes	zxcvbn	03:41.34 s	03:41.34 s
4	Pengujian4. avi	953,876 bytes	zxcvbn	04:10.12 s	04:10.12 s
5	Pengujian5. avi	1,400,836 bytes	zxcvbn	04:25.24 s	04:25.24 s

Tabel diatas merupakan pengujian terhadap waktu dengan jenis dan ukuran file *.avi yang berbeda menggunakan algoritma blowfish dengan panjang kunci 32 bit dan jumlah iterasi 16 putaran.

Tabel 6.6 Pengujian terhadap *file* *.rar

No	Nama File	Ukuran File (Bytes)	Kata kunci (Password)	Waktu Enkripsi	Waktu Dekripsi
1	Pengujian1.rar	197 bytes	mnbvcx	0.16 s	0.16 s
2	Pengujian2. rar	20,262 bytes	mnbvcx	04.22 s	04.22 s
3	Pengujian3. rar	381,556 bytes	mnbvcx	47.06 s	47.06 s
4	Pengujian4. rar	419,160 bytes	mnbvcx	98.78 s	98.78 s
5	Pengujian5. rar	765,814 bytes	mnbvcx	01:02.23 s	01:02.23 s

Tabel diatas merupakan pengujian terhadap waktu dengan jenis dan ukuran file *.rar yang berbeda menggunakan algoritma blowfish dengan panjang kunci 32 bit dan jumlah iterasi 16 putaran.

Hasil pengujian terhadap waktu dengan jenis dan ukuran file yang berbeda Pada file *.Text. *.Doc, *.Pdf, *.ppt, *. Jpg, *.Gif, * Mp3, *.Mpeg, *.Avi, *.Rar menunjukkan waktu yang sama antara waktu enkripsi dengan waktu dekripsi.

5.2.6 Pengujian terhadap *Attack*

Pengujian kekuatan algoritma blowfish dari serangan dengan menggunakan jenis serangan (*attack*) *Exhaustive attack* atau *brute force attack*. Percobaan yang dibuat untuk mengungkap *plaintext* atau kunci dengan mencoba semua kemungkinan kunci (*trial and error*).

5.2.6.1 Batasan Pengujian Terhadap *Attack*

Batasan pengujian terhadap attack pada tugas akhir ini adalah sebagai berikut :

1. Pengujian dalam proses *attack* terhadap algoritma dilakukan sebanyak 50 kali percobaan.
2. Format *file* yang digunakan pada proses pengujian adalah (*.text).

Asumsi yang digunakan dalam pengujian terhadap *attack*:

1. Kriptanalis tidak mengetahui algoritma yang digunakan.
2. kriptanalis memasukan kemungkinan kunci yang digunakan secara acak

Tabel 6.7 Pengujian Terhadap *Attack*

No	Ukuran *file* *(bytes)*	*Password* awal	*Password* Uji coba	Hasil
1	3,878 *bytes*	rud123	123456	Tidak berhasil
2	3,878 *bytes*	rud123	asdfgt	Tidak berhasil
3	3,878 *bytes*	rud123	wer435	Tidak berhasil
4	3,878 *bytes*	rud123	asdqwe	Tidak berhasil
5	3,878 *bytes*	rud123	654321	Tidak berhasil
6	3,878 *bytes*	rud123	678905	Tidak berhasil
7	3,878 *bytes*	rud123	aszxcf	Tidak berhasil
8	3,878 *bytes*	rud123	bgfdrs	Tidak berhasil

9	3,878 *bytes*	rud123	gfgthy	Tidak berhasil
10	3,878 *bytes*	rud123	drfesw	Tidak berhasil
11	3,878 *bytes*	rud123	frtyuj	Tidak berhasil
12	3,878 *bytes*	rud123	sdweas	Tidak berhasil
13	3,878 *bytes*	rud123	asdefr	Tidak berhasil
14	3,878 *bytes*	rud123	asdrfg	Tidak berhasil
15	3,878 *bytes*	rud123	67yt5r	Tidak berhasil
16	3,878 *bytes*	rud123	wedrs1	Tidak berhasil
17	3,878 *bytes*	rud123	aqswed	Tidak berhasil
18	3,878 *bytes*	rud123	gtyhuj	Tidak berhasil
19	3,878 *bytes*	rud123	nmkloi	Tidak berhasil
20	3,878 *bytes*	rud123	hnjuyi	Tidak berhasil
21	3,878 *bytes*	rud123	ghytuj	Tidak berhasil
22	3,878 *bytes*	rud123	78iujk	Tidak berhasil
23	3,878 *bytes*	rud123	mklop0	Tidak berhasil
24	3,878 *bytes*	rud123	hjukio	Tidak berhasil
25	3,878 *bytes*	rud123	drftgh	Tidak berhasil
26	3,878 *bytes*	rud123	swderf	Tidak berhasil
27	3,878 *bytes*	rud123	nbhgtf	Tidak berhasil
28	3,878 *bytes*	rud123	mnjhbg	Tidak berhasil
29	3,878 *bytes*	rud123	nlkjhf	Tidak berhasil
30	3,878 *bytes*	rud123	zxscbt	Tidak berhasil
31	3,878 *bytes*	rud123	tmusbg	Tidak berhasil
32	3,878 *bytes*	rud123	bhwdli	Tidak berhasil
33	3,878 *bytes*	rud123	mklout	Tidak berhasil
34	3,878 *bytes*	rud123	poasde	Tidak berhasil
35	3,878 *bytes*	rud123	cfvtrd	Tidak berhasil
36	3,878 *bytes*	rud123	markiz	Tidak berhasil
37	3,878 *bytes*	rud123	hnbgds	Tidak berhasil
38	3,878 *bytes*	rud123	765tre	Tidak berhasil

39	3,878 *bytes*	rud123	34redf	Tidak berhasil
40	3,878 *bytes*	rud123	98iogd	Tidak berhasil
41	3,878 *bytes*	rud123	876yhb	Tidak berhasil
42	3,878 *bytes*	rud123	0okmnj	Tidak berhasil
43	3,878 *bytes*	rud123	i98uhb	Tidak berhasil
44	3,878 *bytes*	rud123	7ygvcf	Tidak berhasil
45	3,878 *bytes*	rud123	t65rdx	Tidak berhasil
46	3,878 *bytes*	rud123	zse43w	Tidak berhasil
47	3,878 *bytes*	rud123	aq21jk	Tidak berhasil
48	3,878 *bytes*	rud123	Mnbvfg	Tidak berhasil
49	3,878 *bytes*	rud123	Thdbzc	Tidak berhasil
50	3,878 *bytes*	rud123	Pooklm	Tidak berhasil

Aplikasi kriptografi sebagai pengamanan data pada file dengan menggunakan algoritma blowfish memberikan pembuktian kekuatan terhadap serangan (*attack*) dengan menggunakan metode *brute force attack*. Pada pengujian menunjukan hasil bahwa dari 50 kali usaha pembobolan *password*, persentase kegagalan 100 % dan tidak merusak file yang telah dienkripsi.

5.3 Kesimpulan Pengujian

Setelah membandingkan antara hasil perancangan dan hasil yang didapat, maka dapat dilihat bahwa kriptografi pada file menggunakan algoritma blowfish dapat dilakukan dengan sempurna. Tampilan aplikasi yang dihasilkan bersifat *user friendly* ketika diuji coba kepada beberapa *user*. Adapun yang dapat disimpulkan dari beberapa pengujian sebagai berikut:

1. Pengujian dilakukan dengan menampilkan hasil bilangan biner asal dengan bilangan biner hasil enkripsi. Dari hasil pengujian diperoleh hasil bahwa data bilangan biner sesuai dengan proses awal enkripsi maupun dekripsi.
2. Pengujian pada aplikasi ini dapat menghapus bahkan mengubah manual pada extensi penanda file yang akan merusak rantai enkripsi dan dekripsinya termasuk perhitungan bit asli dan hasilnya. Hal ini terbukti dari hasil perubahan pada extensi penanda file setelah di enkripsi dan dilakukan proses dekripsi, maka file tersebut tidak dapat diproses.
3. Pengujian dilakukan sebanyak dua kali proses enkripsi dan untuk mendapatkan file asli maka dilakukan proses dekripsi sebanyak dua kali juga.
4. Hasil dari pengujian terhadap modul berhasil diproses dengan memberikan secara detail mengenai item-item yang terdapat pada setiap tampilan proses enkripsi dan dekripsi file.
5. Hasil pengujian terhadap jenis dan ukuran file yang berbeda berhasil dilakukan karena proses algoritma blowfish bisa dilakukan berulang kali jika bilangan biner pada file melebihi dari 64 bit.
6. Hasil dari pengujian terhadap waktu dengan jenis dan ukuran file yang berbeda menunjukkan perbedaan jika ukuran file yang semakin besar.
7. Pengujian keamanan hasil enkripsi menggunakan metode *brute force attack* menunjukan hasil bahwa dari 50 kali usaha pembobolan *password,* persentase kegagalan 100 % dan tidak merusak file.

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan analisa, perancangan dan implementasi pada aplikasi kriptografi sebagai pengamanan data dengan algoritma blowfish, dapat diambil kesimpulan sebagai berikut :

1 Aplikasi kriptografi sebagai pengamanan data pada file dengan menggunakan algoritma blowfish berhasil memproses enkripsi dan dekripsi pada semua file dengan berbagai pengujian.

2 Algoritma blowfish memiliki kekuatan keamanan yang dapat dibuktikan terhadap serangan (*attack*) dengan mengunakan metode *brute force attack.*

6.2 Saran

Beberapa hal yang disarankan dalam pengembangan aplikasi kriptografi sebagai pengaman data pada file dengan algortima blowfish ini adalah sebagai barikut:

1. Pada aplikasi ini menggunakan semua file sebagai bahan dalam pengujian pada kriptografi dengan algoritma blowfish dan tidak menutup kemungkinan dikembangkan dengan menggunakan folder sebagai bahan

DAFTAR PUSTAKA

Andi. *Memahami Model Enkripsi dan Security Data*, Andi Offset, Jogjakarta,2003.

Ilmukomputer.” Sekilas Tentang Enkripsi Blowfish”URL: 28 Maret 2009 http://ilmukomputer.org/2007/07/27/sekilas-tentang-enkripsi-blowfish/ , 2009

Jethefer, Stevens. *Studi Dan Perbandingan Algoritma Idea (International Data Encryption Algorithm) Dengan Des (Data Encryption Standard)*. Institut Teknologi Bandung: Bandung. 2006

Kadir, Abdul., *Pengenalan Sistem Informasi*, halaman 56-57, CV. Andi Offset (Penerbit Andi) : Yogyakarta, 2003.

Munir, Rinaldi. *Kriptografi*. Institut Teknologi Bandung: Bandung.2007

Rahardjo, Budi. *Keamanan System Informasi Berbasis Internet*. Institut Teknologi Bandung: Bandung.1998

Rudianto. *Analisis Keamanan Algoritma Kriptografi RC6*. Institut Teknologi Bandung: Bandung.2008

Schneier, Bruce.”Blowfish”URL: 2 April 2009 http://www.schneier.com/paper-blowfishfse.html, 2009

Setiadi, Budi. *Analisis Sistem Keamanan Data Dengan Menggunakan Metode Des Dan Metode Gost.* Institut Teknologi Bandung: Bandung.2004

Wahyono, Teguh. *SISTEM INFORMASI (Konsep Dasar, Analisis Desain dan Implementasi)*. Graha Ilmu: Yogyakarta.2004

Wikipedia.” Cryptograpy” URL: 5 April 2009 http://en.wikipedia.org/wiki/Cryptograpy, 2009

Wikipedia.”Blowfish”URL: 6 April 2009 http://en.wikipedia.org/wiki/Blowfish_(cipher), 2009